1985

Evaluation of a data dictionary during application program design based upon the FOCUS DBMS.

Repp, Robert C.

http://hdl.handle.net/10945/21315

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

EVALUATION OF A DATA DICTIONARY DURING APPLICATION PROGRAM DESIGN BASED UPON THE FOCUS DBMS

by

Robert C. Repp

September 1985

Thesis Advisor:	D. R. Dolk
Co-Advisor:	N. R. Lyons

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Evaluation of a Data Dictionary During Application Program Design Based Upon the FOCUS DBMS		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis September 1985
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Robert C. Repp		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943-5100		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943-5100		12. REPORT DATE September 1985
		13. NUMBER OF PAGES 123
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution is unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

data base management systems, data dictionary

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This thesis considers the design and uses of a data dictionary within the FOCUS DBMS package. The thesis details the background and design considerations of an application program for the Director of Admissions at the Naval Postgraduate School. The program will aid in the assignment of an Academic Profile Code (APC) for newly commissioned Naval Officers. A data dictionary is designed and implemented, and its use during application program is development is discussed. Features from (Continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

ABSTRACT (Continued)

commercial DBMSs, fourth generation languages, and data dictionaries are compared and their impact on information systems considered.

Evaluation of a Data Dictionary During Application Program Design Based Upon the FOCUS DBMS

by

Robert C. Repp
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1974

Submitted in partial fulfillment of the
requirements for the degree of

MASTERS OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September 1985

[illegible]
t358
.2

ABSTRACT

This thesis considers the design and uses of a data dictionary within the FOCUS DBMS package. The thesis details the background and design considerations of an application program for the Director of Admissions at the Naval Postgraduate School. The program will aid in the assignment of an Academic Profile Code (APC) for newly commissioned Naval Officers. A data dictionary is designed and implemented, and its use during application program development is discussed. Features from commercial DBMSs, fourth generation languages, and data dictionaries are compared and their impact on information systems considered.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. ASSIGNMENT OF ACADEMIC PROFILE CODES AT THE NAVAL POSTGRADUATE SCHOOL

A. BACKGROUND

At the Naval Postgraduate School (NPS), the Director of Admissions is tasked with evaluating every newly commissioned Naval Officer concerning eligibility for postgraduate education. The evaluation process uses information contained in undergraduate transcripts to compute a quality point rating (QPR). In addition, course subject areas are considered before the assignment of an academic profile code (APC). Approximately 3000-5000 transcripts are processed annually by hand.

The APC is a three digit code that reflects an individual's QPR as well as background in specific mathematical and scientific areas. The first digit of the APC is derived from Table I. The second digit is the mathematics code and is based on the criteria in Table II. The third digit is the science/engineering code and is based on information in Table III.

The Director of Admissions at NPS feels that the APC system is a good representation of a student's technical ability, but does not adequately consider non-technical

TABLE I

FIRST DIGIT OF AN APC

CODE	QPR RANGE
0	3.60-4.00
1	3.20-3.59
2	2.60-3.19
3	2.20-2.59
4	1.90-2.19
5	0-1.89

subject areas. It is conceivable that a fourth digit might be added that reflects non-technical performance.

B. OBJECTIVE

FOCUS has been selected as the Data Base Management System (DBMS) for NPS. It is the objective of this thesis to discuss the design of a FOCUS data dictionary and evaluate its use during the development of an application program to aid in the assignment of APCs.

C. METHODOLOGY

Despite the fact that the DBMS for this application was chosen in advance, this thesis will compare data base models and explore features from numerous commercial systems. It will also look at the concept of dependent and independent data dictionaries. A basic data dictionary will be designed and implemented to aid in the development of

TABLE II

SECOND DIGIT OF AN APC

CODE	MEANING
0	Significant post-calculus math with B or better average
1	Calculus sequence completed with B+ or better average
2	Calculus sequence completed with average between C+ and B
3	Two or more pre-calculus courses with B or better average or one calculus course with C or better grade
4	Two or more pre-calculus courses with average between B- and C+
5	One pre-calculus course with C or better grade
6	No college-level math with C or better grade

FOCUS applications. Design considerations, within the FOCUS environment, are identified for the dictionary and APC application program. An evaluation of the usefulness of a dependent data dictionary during system design will be made.

TABLE III

THIRD DIGIT OF AN APC

CODE	MEANING
0	Significant upper division technical courses with B+ or better average
1	Significant upper division technical courses with average between C+ and B
2	Completed calculus-based physics sequence with B+ or better average
3	Completed calculus-based physics sequence with average between C+ and B
4	One calculus-based physics course with C or better grade
5	No pertinent technical courses

II. DIRECTOR OF ADMISSIONS OFFICE

A. NATURE OF THE PROBLEM

The majority of undergraduate transcripts come from the Naval Academy and a handful of NROTC Universities. The remainder come from colleges and universities all around the United States. Since there is no standard transcript format, it is necessary that each one be reviewed, and pertinent information summarized to assign an APC.

Currently, the Director of Admissions at NPS is calculating QPRs manually and assigning APCs based on hard copy transcripts. Considering the number of transcripts submitted every year, this process is not only time consuming, but costly and error prone.

The Naval Academy has agreed to provide NPS with approximately one thousand undergraduate transcripts annually via computer tape. If this one input were partially automated, it could result in a twenty to thirty percent reduction in workload associated with the assignment of APCs.

The tape consists of transcripts sorted by student ID in ascending order. An example of records contained on the tape is shown in Table IV. The individual records are

TABLE IV

TRANSCRIPT RECORD

```
841029 ABDUL DAVID BROS 512727258 SAS M HISPANIC
SM111 V 081HE111 C 181HH103 C 181NS101*B 181PE101 C 181SC105 B 191
SI100 A 181SM112 B 181NL102*B 281HH104 D 281HE112 B 281EN100*B 281
----- - -------- - -------- - -------- - -------- - -------- - ---
----- - -------- - -------- - -------- - -------- - -------- - ---
SI431 B 284SI412 B 284HH380 C 284NN412 B 284PE407 RD284PCR400C 284
```

separated by blank lines. Each record begins with an identification line. The composition of the line is:

1- 6	student ID number
7	blank
8-23	student name
24	blank
25-33	social security number
34	blank
35-37	major
38	blank
39	sex (M/F)
40	blank
41-48	race or foreign national

After the identification line, the courses are listed, six per line in blocks of 11 characters per block. Each 11 character block consists of:

1- 6	course name
7- 8	course grade
9-11	semester taken

Some special codes are used throughout the file to aid in transaction processing. The grade field uses "V" and "R" as special codes to indicate courses validated or repeated respectively. Valid entries for letter grades in the grade field are listed in Table V.

Additionally, the Naval Academy will provide a computer tape listing of course names and credit hours associated. Course names, grades, and credit hours are required to calculate a QPR for each graduate. An example of the listing, sorted by course name in ascending order (A-Z,0-9), is shown in Table VI.

Most undergraduate universities calculate a Quality Point Rating in the same manner. Nevertheless, two problems arise when a QPR calculated by an outside source is used by NPS to assign an APC. The most straightforward of the two is the conversion from the three point to the four point scale. This is not the case with the Naval Academy, hence the solution will not be discussed in this thesis. On the qualitative side, if an undergraduate student repeats a course and receives a higher grade, only the higher grade goes into the calculation of a QPR. If this fact were ignored in the assignment of an APC, it could mean that a

TABLE V

VALID LETTER GRADES

NORMAL	VALIDATE **	REPEATED COURSE*
A	V	RD
B		RF
C		
D		
F		

* used in QPR calculations ** not used in QPR calculations

TABLE VI

COURSE CREDIT HOURS

EA303,2	EE221,4	EE494,2	FE301,3	FP314,3
HE111,3	HE442,3	HH224,3	HH362,3	NS101*,3
SM201,4	SO471,3	SP440,3	SP493,2	SR401,3
-------	-------	-------	-------	-------
-------	-------	-------	-------	-------

* indicates courses used in Military QPR calculation

student who repeated a course to receive a passing grade could be assigned the same APC as a student who easily completed the course initially. This is not the intention of the APC. To preclude this possibility, all undergraduate QPR's are recalculated by the Director of Admissions at NPS. The new QPR includes all grades received by a student, and in a sense becomes a common denominator for academic evaluation.

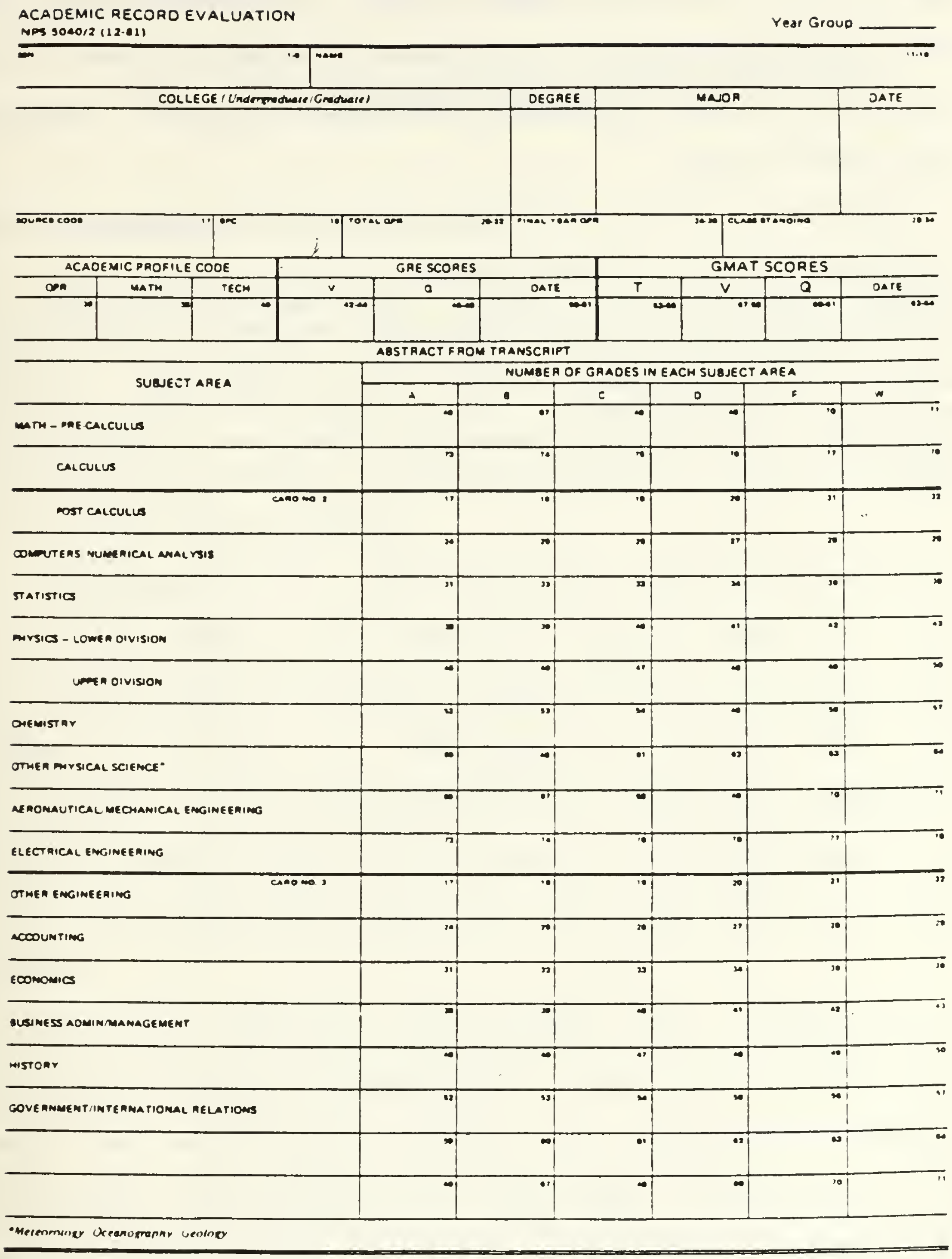
B. SYSTEM REQUIREMENTS

One formal report, shown in Figure 2.1, dictates all system requirements. NPS Form 5040/2 (revised 12/81), Academic Record Evaluation is a transcript summary that facilitates the assignment of an APC. It is also used by the Department Chairman to aid in a subjective judgment on whether an individual's academic background will support a specific masters program. The queries required to complete Form 5040/2 are listed below.

1. Count the number of courses in each subject area by letter grade.
2. Calculate a QPR based on valid letter grade and course credit hour entries.

Before the design of the application program described above, a data dictionary was implemented to aid in program development. The requirements and design specifications for the data dictionary are discussed in Chapter 4.

The decision to purchase and use FOCUS as the administrative DBMS for NPS was made primarily by the Admissions Office programming staff. Their choice was heavily influenced by programming capabilities and a modular pricing policy. Basic FOCUS, excluding options like financial modeling and graphics, can be purchased for less than many other mainframe systems. Nevertheless, it is important that policy and decision makers be involved in the



ACADEMIC RECORD EVALUATION
NPS 5040/2 (12-81)

Year Group ________

SSN 1-9	NAME 11-16

COLLEGE *(Undergraduate/Graduate)*	DEGREE	MAJOR	DATE

SOURCE CODE 17	SPC 18	TOTAL QPR 20-22	FINAL YEAR QPR 24-26	CLASS STANDING 28-34

ACADEMIC PROFILE CODE			GRE SCORES			GMAT SCORES			
QPR	MATH	TECH	V	Q	DATE	T	V	Q	DATE
36	38	40	42-44	46-48	50-51	53-56	57-58	60-61	63-64

ABSTRACT FROM TRANSCRIPT

SUBJECT AREA	NUMBER OF GRADES IN EACH SUBJECT AREA					
	A	B	C	D	F	W
MATH – PRE CALCULUS	66	67	68	69	70	71
CALCULUS	73	74	75	76	77	78
POST CALCULUS CARD NO 2	17	18	19	20	21	22
COMPUTERS NUMERICAL ANALYSIS	24	25	26	27	28	29
STATISTICS	31	32	33	34	35	36
PHYSICS – LOWER DIVISION	38	39	40	41	42	43
UPPER DIVISION	45	46	47	48	49	50
CHEMISTRY	52	53	54	55	56	57
OTHER PHYSICAL SCIENCE*	59	60	61	62	63	64
AERONAUTICAL MECHANICAL ENGINEERING	66	67	68	69	70	71
ELECTRICAL ENGINEERING	73	74	75	76	77	78
OTHER ENGINEERING CARD NO 3	17	18	19	20	21	22
ACCOUNTING	24	25	26	27	28	29
ECONOMICS	31	32	33	34	35	36
BUSINESS ADMIN/MANAGEMENT	38	39	40	41	42	43
HISTORY	45	46	47	48	49	50
GOVERNMENT/INTERNATIONAL RELATIONS	52	53	54	55	56	57
	59	60	61	62	63	64
	66	67	68	69	70	71

*Meteorology Oceanography Geology

Figure 2.1 Academic Record Evaluation Form

specification of system capabilities. A problem arises when management is unfamiliar with common DBMS attributes. It is difficult to specify requirements without being familiar with capabilities.

Chapters 3 and 4 are intended to provide management with a background and understanding of the capabilities of current DBMSs and data dictionaries. A better understanding should allow managers, users, and programmers to take advantage of the fourth generation environment provided by FOCUS.

III. DATA BASE MANAGEMENT SYSTEMS AND FOURTH GENERATION LANGUAGES

A. COMPARISON OF DATA BASE MODELS

Before an attempt can be made to understand, design, implement, or simply use a DBMS, a data model is needed. In addition, not all people think alike, so that different approaches may appeal to different people. Hence, different data models may be needed.

The selection of a DBMS should be heavily influenced by the data models and resulting logical views of data provided. FOCUS is primarily based on the hierarchical data model. It should be determined in advance that data structures, intended applications, and users are compatible with the capabilities and limitations of the data model. If they are, a DBMS that implements that model would be a good choice.

If the underlying data model for a DBMS is not considered before implementation, there is a chance that data may have to be forced into incompatible logical structures to conform with data model limitations. The Administration at NPS should be aware of the strengths and weaknesses of the data model underlying FOCUS and the available alternatives.

Over the past ten years, no fewer than twenty data models have been proposed. Each has its own concepts and terminology. There are three major models currently implemented by commercial systems. The following sections provide an overview of the CODASYL, Hierarchical, and Relational data models as described by [Ref. 1,2].

1. CODASYL

The words schema and subschema were first highlighted in 1969, by the CODASYL Data Base Task Group (DBTG), to describe different views of data. It was the concept of a common Data Description Language (DDL) that gave rise to three specific views of data; the global conceptual view or schema, the external or subschema view, and the internal view or physical data base description.

The schema is the complete, logical view of the database, and subschemas are subsets of schemas as viewed by application programmers. Not all records or relationships of a schema are exposed to subschemas, but when included in a subschema, they can be renamed and data-item formats can be changed.

To aid in the separation of physical and logical views of the database, the schema does not contain the physical description of the database. The Data Structure Definition (DSD) is used to map physical storage requirements and to specify access methods.

The CODASYL model is a physical database model that makes use of data-items, records, and a relationship called a set. A set consists of a two-level tree of records. Each set type can represent a relationship between two or more record types. Each set must contain one occurrence of its owner record and may contain any number of member records (one-to-many). Figure 3.1 gives an example of a CODASYL set.

A multilevel hierarchical file or simple network can be regarded as being composed of multiple sets. Each one-to-many relationship is just replaced by a set. An early criticism of the DBTG proposed DML was its inability to describe complex plex structures also known as networks. This is not necessarily a serious disadvantage because the complex structures can be decomposed to simple networks by defining intersection records. What does suffer, however, is the logical view of the data.

The CODASYL Task Group also addressed important topics such as logical and physical independence vs. response time and data integrity. The implementation of areas was a compromise between too much physical control which destroys data independence and too little physical control which sacrifices response time. An area is a named subdivision of a database. They are intended for use by the DBA to enhance system efficiency through control of record placement.

```
+----------------------------------------------------------+
|                       OWNER RECORD                       |
|                    +---------------+                     |
|                    | record type-1 |                     |
|                    +-------+-------+                     |
|                            |                            |
|                            |                            |
|                            |                            |
|           +----------------+---------------+            |
|           |          MEMBER RECORDS        |            |
|     +------+------+                 +------+------+      |
|     |record type-2|+                |record type-3|+     |
|     +-------------+|+               +-------------+|+    |
|      +-------------+|                +-------------+|    |
|       +-------------+                 +-------------+    |
|                                                          |
+----------------------------------------------------------+
```

Figure 3.1   Example of a CODASYL Set

The CODASYL model addressed data protection by implementing privacy locks. These locks are imposed at any level in the system from the overall schema to an individual data item. Without the proper key, similar to a password, the update/modify type functions can be inhibited.

When the notion of a standard began to gain support, so did the controversy over what the standard should be. IBM's Data Language/I, a hierarchical based language, along with proposed relational systems began to challenge CODASYL for the title of industry standard.

2. Hierarchical

IBM began the development of Data Language/I (DL/I) in the 1960's to support the aerospace industry. It is still the basis for their DBMS called Information Management System (IMS).

In DL/I, a field is the smallest unit of data. A group of fields is known as a segment and a hierarchically structured group of segments constitutes a data base record. Each relationship between records is one-to-many with all arcs pointing toward the leaves. At this point, the similarities between the CODASYL logical set and DL/I hierarchical structure can be seen. Figure 3.1 could be relabeled and used for the hierarchical model. As a result of this similar tree structure, DL/I must transform network relationships into hierarchical form in much the same way as the CODASYL sets. The network is first decomposed into trees with duplication, and then logical pointers are used to remove the redundancy. These logical child pointers make it possible to represent multiple logical structures from any number of physical trees. Figure 3.2 gives an example of the process.

The three views of data described earlier, are not supported by DL/I or the FOCUS Data Description Language. The only distinction made is between logical and physical data base records. A logical data base may be either a subset of a physical data base, or it may contain parts of two or more physical data bases. Segment relationships can be present in the logical structure that do not exist in the physical data base.

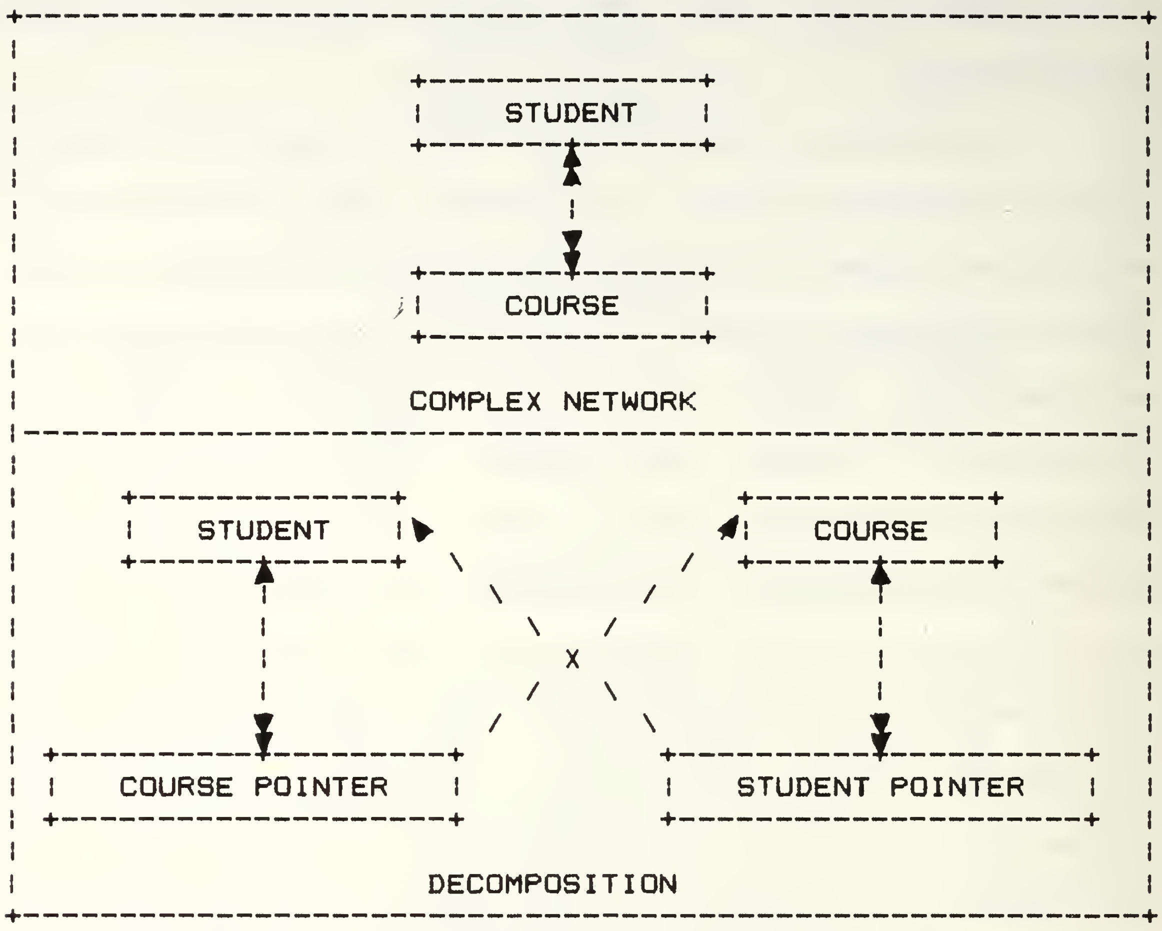


Figure 3.2 Decomposition of a Complex Network

Sensitivity is a term used to describe the ability of application programmers to maintain separate views of the same data base. Application programs are sensitive only to changes made in the physical data base that are reflected in their logical view. In some cases, the physical data base may be modified without changing application programs. In other words, the logical view is insensitive to unrelated changes in the physical data base. The separation of logical and physical views is an important step toward data independence.

3. Relational

Dr. E. F. Codd first introduced the relational model in 1970 [Ref. 3]. As early as 1975, this model was being heralded as the key to further automation in database management software. Nevertheless, it was not until the early 1980's that a commercially viable relational DBMS was available. The first was IBM's SQL/DS, followed closely by ORACLE from Relational Software Inc. Both systems use Structured Query Language (SQL) for query, data manipulation, and data definition.

The relational model is unique in that it is used only for the logical description of data. A relational database can be physically structured many different ways. No knowledge of the underlying data structures such as linked or inverted lists is required. Physically oriented models can inhibit changes to data that may be needed as the database grows, because the changes would dictate expensive application program modification.

Two-dimensional flat files, called relations, are used to represent data in the relational model. Columns are called attributes and rows are called tuples. Each attribute must have a unique name and a domain, the set of values that the attribute can have. A tuple is said to be of degree n, where n is the number of attributes. No two tuples in a relation can be identical and their order is insignificant. The generalized form of a relation and several occurrences are depicted in Figure 3.3.

STUDENT RELATION

STUDENT(ID#, name, age, sex, major)

747231	SMITH	18	M	EE
747233	JONES	17	F	ENG
747722	MADISON	17	F	PHY
744345	JOHNSTON	18	M	MTH

Figure 3.3 Example of Relational Flat Files

To insure a flexible and consistent logical representation, Codd designed a technique called normalization. First normal form involves the elimination of repeating groups by specifying that each attribute must be a fact about the key. Second normal form reduces redundancy by insuring that every non-key field is a fact about the entire key. Third normal form eliminates deletion anomalies by specifying that a non-key field cannot be a fact about any other non-key field. With normalization and intersection records, any representation can be reduced to two-dimensional flat files with some redundancy.

To provide a subschema that is logically separate from the physical database, it is necessary to be able to extract subsets of rows and columns from the relations that make up the schema. The name used to describe this manipulation of relations is relational algebra. Using relational algebra, operators are defined that work on relations. The practical use of relational algebra is seldom seen because of its procedural structure, but the underlying concepts are important.

The advantage of a nonprocedural language is that the user specifies what is required, possibly in English-like terms, and the system can optimise how to obtain the required data. Nonprocedural language interfaces have taken many forms. For example, SQL is a nonprocedural transfer-oriented language that is capable of using relations to transfer data into results.

The relational model is the most widely accepted and adopted in commercial applications today. The individual reasons vary from its mathematical basis to its logical structure. The bottom line is that the user, whether casual or experienced programmer, can understand and maintain the resulting logical structure much more easily than with the previously described models.

In a small database, with minor applications, there may be no advantage for relational over network or hierarchical. However, as the database grows and data and relationships are added and changed, a properly designed relational system often is much simpler and less expensive to maintain. Good data independence and normalized form will usually allow growth and change in the database without forcing the rewriting of application programs. Without these characteristics, the cost of maintaining an organization's programs and data can quickly rise to an unacceptable level.

B. FOURTH GENERATION LANGUAGES

NPS is still new to the data base environment and already a programming backlog has developed. The delay for non-trivial applications can exceed six months. Possible solutions include hiring more programmers, increasing existing programmer productivity, and involving users in application proggram development. The fourth generation environment provided by FOCUS is capable of increasing productivity and involving users.

The evolution of computer languages has occurred in distinct stages. Machine language, patterns of ones and zeros, was the first followed by second generation assembly language which used symbols and mnemonics to replace the patterns of zeros and ones. Third generation, high-level procedural languages such as COBOL, FORTRAN, and Pascal were next to evolve. With each stage in the development of human-computer communication, the programmer was required to know less about the physical composition of data structures.

The concept of data independence, where the programmer is not concerned with the structure of data, was an early goal of data base designers. It was not until the advent of the relational data base that this goal was realized. As a result of its data independent design, the relational data base facilitated progression to the next stage in computer language development, the fourth generation language (4GL). Richard Cobb, one of the developers of the first 4GL said, "In fact, without0 structure-independent DBMSs, fourth generation languages would never have come into existence." [Ref. 4:p. 92]

There are two main differences between third and fourth generation languages. First, the number of programmed instructions is typically less when 4GLs are used. But, it is the nonprocedural structure of 4GLs that make them unique. A user tells the computer what he wants rather than how to retrieve it. This lends itself easily to English-like command structures.

Productivity is the focus of most comparisons between third and fourth generation languages. The tendency is to perform some quantitative analysis, based on historical and projected programmer productivity, to support the transition to fourth generation operations. Richard Cobb, developer of Ramis II, computed a savings of approximately $1,000,000 with a 4GL versus only $23,000 with a 3GL. Very little effort was devoted to justifying the remaining 4GL benefits. The most important is that it is easily used by both computer specialists and end users. [Ref. 4:pp. 91-97].

Very few people dispute the increase in individual programmer productivity with 4GLs, although some have been quick to point out other weaknesses, such as the lack of an industry standard. The development of computer software is moving so quickly that it is difficult, at times, to determine which advance is a step and which is a landing. For that matter, it is hard to tell which development is an advance. Claims that organizational productivity bog down because of the impact of a 4GL product on the existing environment is more than likely a result of the environment and not the 4GL. Without question, systems and organizations will have to change to take advantage of fourth generation application development capabilities. It is not unusual for technology to dictate change. Organizations that do not change, but rely on technology alone to increase their overall productivity, may be disappointed with 4GLs. [Ref. 5:pp. 99-104]

C. COMMERCIAL DATA BASE MANAGEMENT SYSTEMS

There are dozens of commercially available Data Base Management Systems. Each one has relative strengths and weaknesses. It is up to the system designer to determine requirements, evaluate the available systems, assess future trends, and select the most appropriate.

Applications and systems designers may suffer from a lack of flexibility in the modeling of data relationships due to constraints imposed by a particular DBMS. They are forced to compromise in their selection of data models. In some cases, applications are so diverse that multiple data models are employed, necessitating the use of more than one DBMS.

According to Has Patel, Manager of Product Planning for MSP Inc., the solution is to plan an environment in such a way that the DBMS is separated from the application system. He asserts that the emphasis should be placed on Logical Data Structure Designs (LDSD) which are independent of physical DBMS implementations. This could be accomplished with a data dictionary as a tool for the management and control of a corporate data resource. [Ref. 6:pp. 86-91]

A DBMS should perform the following functions:

* Define and store the data base structure using a data definition language.

* Load data into the data base from terminals or external data files.

* Provide a wide variety of access methods such as sequential and keyed. Promote logical/physical independence by automatically updating the physical data base when changes are made to the logical structure.

* Provide multiple views of data depending on class of user.

* Provide security features to control access to data

* Enable control over concurrent operations to allow read/write access to multiple users.

* Facilitate backup and recovery with rollback and rollforward utilities.

* Provide data manipulation capabilities via query language.

* Provide report generation capabilities.

To aid in the selection of an appropriate DBMS, short summaries with applicable comments on each of twenty-five commercial systems are provided in the next sections. Table VIII lists vendor information and data model basis for a number of popular DBMSs. The "self-contained" designation in Table VIII signifies that the DBMS is based on its own internal model. These self-contained models usually consist of some combination of the three major data models discussed earlier.

1. Accent R

ACCENT R is a relational DBMS marketed by National Information Systems, Inc. It is designed to run on Digital Equipment Corporation DEC system-10/20 under the TOPS-10/20 or VMS operating system.

TABLE VII

DATA BASE MANAGEMENT SYSTEMS

DBMS NAME	VENDOR	MODEL
Accent R	National Information Systems 20370 Town Ctr. Ln., Suite 130 Curpertino, CA. 95014	Relational
ADABAS	Software AG	Self-Contained
Condor	Condor Computer Corporation 2051 South State Ann Arbor, MI. 41804	Relational
Data Management IV	Honeywell Information Systems	CODASYL
Datacom/DB	Applied Data Research, Inc.	Relational
Dbase II	Ashton-Tate 10150 Jefferson Blvd. Culver City, CA. 90230	Relational
DBMS-990	Texas Instruments	Self-Contained
DBMS-Prime	Prime Computer Incorporated	Self-Contained
DMS 1100	Sperry Computer Systems	CODASYL
DMS-170	Control Data Corporation	CODASYL
DRS	Advanced Data Management 15-17 Main St. Kingston, NY. 08528	Self-Contained
Encompass	Tandom Computers, Inc.	Self-Contained
Express	Management Decision Systems	Relational
Focus, PC/Focus	Information Builders, Inc. 1250 Broadway New York, NY. 10001	Self-Contained
IDIS	Intel Corporation 3065 Bowers Avenue Santa Clara, CA. 95051	Relational

TABLE VII

DATA BASE MANAGEMENT SYSTEMS (CONT.)

DBMS NAME	VENDOR	MODEL
IDMS/R	Cullinet Software, Inc. 400 Blue Hill Dr. Westwood, Massachusetts 02090	Relational
Image	Hewlett Packard 19447 Pruneridge Avenue Cupertino, CA. 95014	Network
IMS	IBM Corporation	Hierarchical
Info	Henco Software Inc. 100 Fifth Ave. Waltham, MA. 02154	Relational
Ingres	Relational Technology, Inc. 2855 Telegraph Ave. Berkeley, CA. 94705	Relational
Inquire	Infodata Systems Inc. 5205 Leesburg Pike Falls Church, Virginia 22041	Self-Contained
MDBS I	ISE-USA 85 West Algonquin Road Arlington Heights, IL. 60005	CODASYL
MDBS III	ISE-USA 85 West Algonquin Road Arlington Heights, Il. 60005	Self-Contained
Model 204	Computer Corporation Of America 4 Cambridge Center Cambridge, Massachusetts 02142	Self-Contained
Nomad2	D&B Computer Services 187 Danbury Road Wilton, CT. 06897	Relational
Oracle	Oracle Corporation 3000 Sand Hill Road Menlo Park, CA. 94025	Relational

TABLE VII

DATA BASE MANAGEMENT SYSTEMS (CONT.)

DBMS NAME	VENDOR	MODEL
Ramis II	Mathematica Products Group P.O. Box 2392 Princeton, New Jersey 08540	Self-Contained
Rapport	Logica, Inc. 3 Embarcadero Center San Francisco, CA. 94111	Relational
Relate/3000	Computer Resources, Inc. 5333 Betsey Ross Dr. Santa Clara, CA. 95054	Relationalal
Rubix	Infosystems Technology 6301 Ivy Lane Greenbelt, MD. 20770	Relational
Seed DBMS	Seed Software/United Telecom 2300 Walnut St., Suite 734 Philidelphia, PA. 19103	CODASYL
SQL/DS	IBM Corporation	Relational
System 1022	Software House Cambridge, MA. 02138	Relational
System 2000	Intel Systems Corporation 3065 Bowers Avenue Santa Clara, CA. 95051	Self-Contained
The Knowledge Manager	Micro Data Base Systems, Inc. Box 248 Lafayette, IN. 47920	Relational
TIS	Cincom Systems, Inc. P.O. Box 11189 Cincinnati, Ohio 45211	Relational
Total	Cincom Systems, Inc. P.O. Box 11189 Cincinnati, Ohio 45211	Self-Contained

The storage structure consists of either fixed or variable length records. All indexes for a given data file are combined into one independent index file. Records may be accessed through keyed or unkeyed fields and RAM indexes provide fast keyed retrieval.

Data entry is controlled by commands that allow input to chosen fields, prompting, duplication of values, pre-testing for errors, and validation based on a data file. Field names are up to forty characters long and are of types integer, real, double precision, character, alpha, date, bit, and user defined.

Accent R provides a non-procedural natural query language, a high-level structured programming language, and host language interfaces for FORTRAN, COBOL, and BASIC.

The Data Base Library is an actively maintained dictionary. It provides information about how programs and data are structured, processed, and related. It works with a utility named SOMOD to recompile all necessary programs when a change is made.

2. Adabas

Adabas is comprised of the Associator (inverted lists, coupled lists and system file), Data Storage, and Work Storage (temp space). Numerous utilities are used to load the data base, add new fields, add new paths, define new relationships, and sequence files without resorting.

The data base files are described to the data dictionary which contains information about the characteristics of data items, such as their relationship to other items and where they are used.

A comprehensive backup and recovery facility logs all changes to the data base and writes coordinated checkpoints. Partially completed transactions are removed and the data base can be restored to any checkpoint if a failure occurs.

Host language calls to ADABAS are embedded in COBOL, FORTRAN, PL 1, and Assembler. ADASCRIPT+, ADACOM, and NATURAL data manipulation languages are provided.

3. Datacom

Datacom is designed for use with IBM mainframe equipment and operating systems. System utilities provide high-speed loading, unloading, and recovery. DATACOM offers complete transaction backout, roll-forward, and roll-backward restart and recovery capabilities. Data base files may be recovered or restored from any point in time.

Record locking is automatic at the logical record level. A program may have many records locked concurrently. Data contention situations are reported to the user in the data base statistics. Deadlock is detected and resolved by the system.

A DATADICTIONARY facility describes the characteristics of data items within the data base. Their relation to other items, applications, and involvement in the entire system are also included. This is an active dictionary facility that contains the control data used to drive Datacom.

Data manipulation may be accomplished using application programs written in either COBOL, PL 1, ALC, FORTRAN, or RPG. A high level language facility (ADR/DL) is available to define and manipulate data using English-like statements. IDEAL is a self-contained relational 4th-generation language. DATAQUERY is a self-contained relational query language.

4. Dbase II

A microcomputer based relational DBMS, dBase II requires CP/M 2.2 or later for 8 bit systems or PC-DOS, MS-DOS, or CPM 86 for 16 bit systems. Files are stored as fixed length ASCII data. Access is sequential or random using inverted B-tree indexes.

The query language allows memory variable storage to be saved to disk, and restored later. Access, security, and validation can also be accomplished through use of the query language. The on-line environment allows the user to access any of several records, edit them or browse through the data base. No active dictionary support is currently available.

5. DBMS - Prime

DBMS - Prime is a CODASYL based system. It supports CODASYL DDL, COBOL, FORTRAN, and CODASYL DML commands. Sub-schema compilers for FORTRAN and COBOL generate object sub-schema files and a DML preprocessor converts CODASYL DML statements to host language calls.

A schema editor is available to support revising the schema in an interactive mode. The stored data base is automatically revised to correctly reflect any changes to the schema. Whenever a sub-schema is recompiled, the system insures that all associated programs are recompiled.

Data integrity is preserved through extensive failure protection procedures. An automatic rollback to the end of the last completed transaction on program failure, rollback utility on system failure, and rollforward utility for system or media failure are just a few.

6. Data Manager IV

Honeywell Information Systems (HIS) markets a comprehensive on-line CODASYL based DBMS named DATA MANAGER IV (DMIV). As with most CODASYL implementations, data base integrity is provided by before and after images, rollback and rollforward functions, and checkpointing. Every sub-schema must be validated against the schema and a user generated DML permitted/prohibited list.

Data base initialization and restructuring are accomplished through utilities. Initial and subsequent data loading are via COBOL or FORTRAN user programs (no preprocessor required). Any record type may be defined as owner or member in any number of sets. This provides hierarchical, tree, and network capabilities. If desired, the entire data base may be viewed as a relational, table-oriented data base. In support of this, HIS provides IQ, EQ, and RQ, three new relational languages.

HIS also provides a Data Dictionary/Directory System (DD/DS) that supports up to eighteen different entity types and their relationships. It can produce fifty-eight unique reports.

7. DMS 1100

Marketed by Sperry Computer Systems, DMS 1100 is a CODASYL DBMS for use with all Sperry series 1100 computers. Item descriptions are based on COBOL-oriented CODASYL specifications including level number, name, picture, usage, occurs, and occurs depending on clauses. Item definitions can include a CHECK clause for data validation.

Data base creation and input are accomplished via user application programs or with a special "load" schema. Modifications require some reloading and recompilation of programs. The Data Dictionary System (DDS), a dependent, active data dictionary, can aid data base revision with impact reports.

Application programming can interface with COBOL FORTRAN and PL/1. DMS 1100 provides two other data manipulation languages. QLP 1100 is a command-oriented data manipulation language and RPS 1100 is a form-oriented screen image report language. Both languages supplied are designed for use by the end user.

Automatic and utility based roll back and recovery facilities are provided. Roll back can affect only individual users or programs, or all active programs at the time of system failure. Selective file recovery is possible and an entire data base may be recovered to a pre-established recovery point.

8. DRS

DRS is a powerful self-contained, relational-like DBMS produced by Advanced Data Management. It is designed to operate with most DEC VAX and PDP systems, as well as IBM mainframes. A multi-processor version of DRS for VAX allows a data base and its users to be spread across several machines.

There are two types of files supporting the physical structure of a DRS data base. The first is the Record Address file that tracks record location by file and page. A version of this bit map marks the locations of qualifying records after a retrieval so they do not have to be copied to temporary storage. Secondly, indexes are maintained through inverted files in a B-tree format. Any field or combination of fields can be indexed. Even partial field indexing, where a single word or phrase in a field is indexed, is available. The storage area for records can consist of up to 1000 logically concatenated physical disk files. Each file is divided into fixed length units called pages. The structure and attributes of each data base are stored in a central dictionary data base. They are accessed via an interactive display-oriented utility called Data Base Builder.

Concurrent read/write access to a data base is available for up to 64 users. Concurrent read only access is unlimited. Automatic locking occurs at the record level while a record instance is actually undergoing update.

DRS is an English-like, self-contained language for query and update. XBS and SIP are the host-language interface and forms-oriented query and update language respectively. Link modules and XBS programs may be written in any standard compiled language.

9. Encompass

Tandem Computers, Inc. have developed a self-contained DBMS for use on all Tandem NonStop systems. It will allow data and applications to be distributed across a 255-node network. A unique relational data base access method named ENSCRIBE ensures that no single failure will result in corruption or loss of data. In fact, failed components may be serviced while the rest of the system remains in operation. The Transaction Monitoring Facility can, if a failure occurs, back out an entire transaction across all nodes of a distributed system. Data bases are reconstructed by rolling-forward from on-line dumps and audit trails.

The data dictionary plays an important role for ENCOMPASS. The on-line dictionary, built with the Data Definition Language Processor, provides a view of relational data bases. Under the purview of the dictionary, these data bases can be queried by ENFORM, the query/report writer language, or validated/updated by PATHWAY, the transaction processing system. ENFORM, with the data dictionary, supports field level security by allowing multiple dictionaries to describe the same data base. Only those fields explicitly mentioned in the dictionary are accessible to the user of that dictionary.

10. Express

Management Decision Systems, Inc. claim that EXPRESS, their relational DBMS, offers the widest range of decision support system tools using a single command syntax. These tools include data management, reporting, graphics, simulation, ad hoc query, statistics, and modelling. EXPRESS is available for IBM mainframe and Prime 400 or larger computers. A large library of statistical, plotting, financial, and market modeling routines along with the ability to easily use and create new EXPRESS commands makes this DBMS user friendly.

The data base is a collection of multidimensional variables. The user may work with up to 256 dimensions at one time and may establish hierarchical relationships between any dimensions.

The system is designed primarily for interactive processing although batch may be accomplished by storing commands in a file. Concurrent read/write access is not provided by the system. Read only access by multiple users is possible.

11. Focus

Focus and PC Focus are manufactured and sold by Information Builders, Inc. They are designed to operate on the IBM 370 and IBM PC (minimum of 5MB external storage) respectively. As with any hierarchical system, many-to-many relational-like structures can be designed but present complicated logical views. Focus is the Administrative DBMS at NPS. More information can be found in Chapters 4 and 5.

12. <u>Intel's Database Information System (IDIS)</u>

IDIS is a standalone, relational, micro-system package with mainframe access capability to SYSTEM 2000. Mainframe data download statistics are provided before actual local database load to prevent out of memory errors.

Intel provides spreadsheet and graphics integration, a word processing interface, and several interactive menu driven utilities. A data dictionary is integral with IDIS and a local dictionary driven data download from mainframe is included.

This is one of the DBMSs that makes full use of the UNIX (or XENIX by Microsoft) operating system. IDIS provides access to XENIX file structure and word processing interface for easy cataloging as well as the XENIX mail and calendar facilities.

13. <u>Integrated Database Management System (IDMS)</u>

IDMS was developed by Cullinet Software for IBM 360/370 series computers. A dependent data dictionary called Integrated Data Dictionary is also available. In combination, these two constitute a powerful CODASYL implementation of a DBMS. A relational version called IDMS/R is also available.

Data description uses standard CODASYL set relationships. Data manipulation is through COBOL, PL/1, Assembler, CALL, or FORTRAN. An option is available that allows a multitasking environment and ensures that more than one task does not update the same record at the same time.

Applications programming is through call statements generated by a preprocessor. Operators and functions are limited to those provided by the host language.

14. IMS

Up to 31 on-line application programs may access an on-line data base using IBM Corporation's DBMS named IMS. Since IMS is designed for IBM mainframe computers, a heavy emphasis is placed on access efficiency, security, and control.

Special storage techniques are used to minimize storage requirements. Free space can be reserved in advance to allow later insertion of segments near their parents or insertion of new root segments. Deleted segment space can be reused for new data.

IMS provides automatic logging of all changes to any data base. It also contains recovery utilities for restoration without re-execution of application programs.

Program Isolation Trace, a utility provided by IMS, shows locking and deadlocking conditions. IMPARS, a productivity aid, can be used to report internal response time and resource utilization. Dictionary support is available through DB/DC Data Dictionary Systems, an IMS dependent data dictionary package also by IBM.

IMS is best suited for real world data applications which exhibit hierarchical relationships.

15. Info

Info is a mini-computer oriented, relational data management system produced by Henco Software Inc. An integrated data dictionary is used to describe logical records to a file and is stored separately from the data base. By creating a dictionary file, INFO can access files already existing in the system. Minimum input validation is also accomplished by the data dictionary.

A combination of a user-friendly language called INFO and multiple add-on products, make INFO appealing to end users. INFO-Veisagraph provides full business graphics linked to data files. INFO-Text joins unstructured data created by a word processor to INFO data files. Also, a financial planning and modeling system, called MODEL, can automatically pull data from INFO to the correct model row and column.

The programming staff is assisted by INFO-Call which links INFO to programs written in a compiled language. INFO-Flow automatically documents INFO application programs.

16. Ingres

A DEC mini or 68000 based micro computer with five megabytes of secondary storage is all that is required to run INGRES, a relational DBMS by Relational Technology, Inc.

As with most relational systems, relationships are established through tables. QUEL, a non-procedural query language, can be used interactively or embedded in BASIC, C, COBOL, FORTRAN, or PASCAL.

Query-By-Forms, Report-By-Forms, and Graph-By-Forms are all designed to free the user from systems programming. Ad hoc queries, reports, and graphics can be obtained without formal programming. Application-By-Forms is an integrated application development system used to accelerate the programming process.

17. Inquire

A modified relational structure was used by Infodata Systems Inc. to design INQUIRE for IBM mainframe computers. In this modified structure, each record entity can be made up of a flat, single entry "owner" element with repeating "member" groups. Similarity between this modified relational structure and the hierarchical model is useful in that hierarchical structures are a byproduct of this approach. It could just as easily have been called a modified hierarchical structure with network relationships as a byproduct.

A single command language for query, report, and data maintenance provides system continuity. Custom macros can be used to create additional commands.

18. <u>The Knowledge Manager (Knowledgeman)</u>

Micro Data Base Systems, Inc. developed KNOWLEDGEMAN, a relational DBMS, for use on the IBM PC.

KNOWLEDGEMAN structured programming language can be used to invoke all data management commands. It supports thirty built-in functions as well as if-then-else, do-while, and case logic. In addition, it is capable of output conversion to ASCII, BASIC, or DIF format.

During data base creation, the user is prompted for values via a standard or customized CRT form. At that time, multiple indexes can be defined for each relation. Index keys can be made up of multiple fields or expressions involving multiple fields. KNOWLEDGEMAN allows virtual fields which require no storage space, as they only exist during execution. Virtual fields may also be defined by expressions involving other fields.

A limited distributed capability is provided by a facility for downloading MDBS III DBMS data files into KNOWLEDGEMAN for local use.

19. The Micro Data Base System III (MDBS III)

Post-relational is how ISE-USA, the creator of MDBS III, classifies their DBMS. MDBS III supports hierarchical, relational, and CODASYL views as subsets of its capabilities. One-to-one, one-to-many, many-to-many, recursive one-to-many, recursive many-to-many, and various other relationships are directly represented by name in MDBS III. This allows a relational join to be accomplished by simply specifying relationship names. Also, virtual views are supported that lack the access and storage inefficiencies of actual tables.

MDBS III provides a query language that is comparable to SQL (by IBM) and an interactive data manipulation language that can be embedded in over twenty host languages. A data dictionary is integral with the system and can be accessed through the query or data manipulation language.

In addition, MDBS III provides several end user oriented applications. Screen Master is an I/O management system. The Report Definition Language (RDL) along with the RDL Analyzer allow a non-programmer to specify the format of a desired report. RDL can also automatically generate C source code for the requested report.

20. Nomad2

Nomad2, a relational DBMS by D&B Computer Services, is designed for on-line (IBM mainframe) interactive problem solving by end users. It employs an English-like non-procedural command language for reporting and updating. Relational and multipath hierarchical data base structures are also supported.

The data management facility allows for easy creation and maintenance of data base files. Data validation spans the range from discreet values and sets to complex logic that may be made a part of the data base description. The SLIST facility provides integrated data dictionary functions such as access to names and characteristics of data items.

The schema may be modified or reorganized without dumping and reloading data. System checks ensure that no changes are made until their validity is verified. Hierarchical segments may also be added, provided they do not interrupt an existing path.

Nomad2 is designed as a shared data base facility which directly supports multiple concurrent write access. Conflicting updates are handled automatically and deadlock situations are prevented by the system.

Numerous other aids such as an automatic procedure generator and an interactive debugging routine make NOMAD2 a stand-alone applications development tool.

21. Oracle

Oracle was one of the first relational mainframe packages to be commercially available on a micro computer. It is highly portable and flexible enough to support ad hoc queries as well as serious application programming.

Oracle Corporation implemented SQL Plus, an extended version of IBM's SQL/DS, as the standard interface to ORACLE. SQL serves as a query, data manipulation, and data definition language and provides full dictionary support. Host language precompilers are available for most languages with embedded call statements.

Normalized, two-dimensional tables are used to represent all data. Access methods are determined dynamically by available inverted keys, physical sequences, uniqueness characteristics, and data distribution.

22. Ramis II

Mathematica Products Group markets Ramis II as the first fifth generation software product for use on IBM mainframe and compatible hardware. Hierarchical, network, and relational file structures are supported. The natural language processor, Ramis II English, is based on the principles of artificial intelligence and can understand and answer queries made in everyday English.

Reporter, a non-procedural language, is used to produce tabular reports. Application programming can be done in COBOL, PL/1, Assembler, and FORTRAN through calls to the data manager.

Failure protection is provided by transaction log files. Access security and control are accomplished through passwords and encryption at all levels from data base to individual fields.

A number of special utilities are also provided that make Ramis II a portable and complete application development environment. REF extends access to non-Ramis files such as Adabas, DL/1, IDMS, IMS, and Total. RAMASTER is a file dictionary that contains information about each field. FSM allows user-defined screen formats of up to 99 frames to be created. RAMLink provides a PC-to-mainframe link.

23. Structured Query Language / Data System (SQL/DS)

SQL/DS, by IBM, is setting the standard in industry for a relational DBMS. It is designed for use on IBM System 370 computers.

SQL, the system language, can be embedded in COBOL, FORTRAN, PL/1, Assembler, or used solely in an interactive mode. It is the most imitated aspect of SQL/DS. Most other system functions such as data base creation, failure protection, and input validation provide adequate capabilities but are often overshadowed by more recent DBMS developments.

24. <u>System 2000</u>

IBM, UNIVAC, CDC, and CYBER computers are all compatible with System 2000, a DBMS produced by Intel Systems Corporation.

System 2000 supports linear, hierarchical, network, and relational logical structures. Each data base is composed of up to six direct access tables such as the integrated data dictionary, indexes, and raw data. Initial data base loading may be accomplished one at a time or via PLEX (Programming Language Extension), a high speed load utility .

The Integrated Data Dictionary plays an important role in System 2000. For each physical data base, the dictionary parameters include archives, update journals, and before-image logs. All internal restructuring is done automatically by the Basic Data Dictionary Language. It also provides password based security down to the individual item level. Input data can be validated for size, type, and picture specified by the schema definition. Additionally, report requests may be made from catalogued procedures stored within the Basic Data Dictionary.

System 2000 offers several, powerful, user-oriented interfaces. Quest is a free-form query and update language. Genius is a conversational report writer and syntax generator. QUEX (Query/Update by Example) is a menu driven formatted screen query and update system. Lastly, System 2000 On Line Operations (SOLO) offers menu driven access to all previously mentioned interfaces plus Report Writer, Tell-A-Graf (graphics package), and the Data Definition Language.

25. Total

In a survey conducted by Cincom Systems, Inc., Total was shown to be integrated into an average of 41% of all user applications. This makes Total the most widely used DBMS by a margin of almost two-to-one over the next leading system. Total operates on 28 different computers on 40 separate operating systems. It is designed to accommodate future hardware changes and promote distributed processing.

Data base creation can be done via user application programs or optional data base administrator utilities. Access methods are dependent on hardware. Adding new elements or paths requires data base regeneration but not necessarily program modification. Application programming can be done using COBOL, PL/I, or FORTRAN.

Two recovery types, point-of-failure and full system, can be accomplished by applying before images in reverse order to the latest checkpoint. Full DBA capabilities to control user access includes subschema which specifies user password, usable data item names, and file access. Deadlock is controlled by a DBA specified time-out.

IV. THE DATA DICTIONARY

A. CONCEPT

By nature, database processing involves the sharing of resources. A DBMS has the advantages of space savings, increased processing speed, and enhanced data integrity as a result of this sharing. Nevertheless, as a database grows, many of the advantages can be negated by poor programming discipline and data standards. If programmers continue to code unnecessary new procedures with unique element names for each new application, the database can suffer from data redundancy. This can cause a system to become more difficult and expensive to maintain. This shared data must be protected through standardization to ensure data integrity and maximize management control. At NPS, the only standards are those imposed by the programmers themselves. A standard name, format, and relationship for every entry in the data base must be established. This is the basic premise behind a data dictionary. It is simply a way to accumulate, update, and report information about data. As a minimum, the data dictionary should give the names, type, length and description of all data items in use. Users should be restricted to only those representations in the dictionary. If a piece of data is defined in more than one area, a need for multiple processes to update this data arises.

Initial implementation of a data dictionary may require extensive data modification. Many large data base systems have several thousand unique data items. The inter-departmental coordination required to standardize, define and name these data items is difficult at best. The Data Base Administrator (DBA) is responsible for setting standards and often encounters resistance from programmers and users. The longer NPS delays in appointing a formal DBA with authority to establish and maintain standards, the harder the process will become. In the initial stages of standardization, the programmers and users can feel limited or hindered in their normal routine. They must learn a new set of rules and comply with them. In their minds, these new standards are just something to make the DBA's job easier at the expense of their own. Sometimes, their feelings are somewhat justified. If the data dictionary is simply used to store information about data without rigorously enforced data administration standards, the only advantage will be the ability to produce reports that document redundant processes and data.

The goal of a data dictionary is not merely to document all the data items contained in a database, but to allow access to all corporate knowledge. A data dictionary can span programs and databases. It may include entity types from fields and formats to memos and relevant publications. It provides programmers and users with a bridge to facilitate communication during application program development. By simplifying the development process, the user can become more involved in system design, thereby reducing the demand on programmers and programming backlog at NPS. [Ref. 7:pp. 89-95]

A data dictionary is in fact a data base about data bases. Consequently, it must either be dependent on a DBMS or contain many of the same components. Among the most important components of a DBMS as well as a data dictionary are the languages necessary for query, data manipulation, and data definition.

* Data Definition Language (DDL). Language for defining type, format and length of data dictionary entries. Data entry validation may be based on definition.

* Data Manipulation Language (DML). English-oriented, on-line or batch language for inserting, changing and deleting entries and relationships. Also used for accessing and obtaining reports of entities and related entities. A set of macroinstructions or call statements provided by a particular DD/DBMS.

* Query Language (QL). User oriented DML directed mainly at interactive ad hoc requests.

* Utilities. Commands for loading, recovery, source language code generation, interfacing to other software products, scanning source code for data dictionary entries, etc.

* Report Writer. Formatting commands for high volume printed reports of entries and related entries.

* Host language Interface. Allows the programmer to access DBMS files through calls in a host language.

* Communication. Link between micro and mainframe for the purpose of sharing data.

While these high level languages constitute the data dictionary system, information about data called metadata is what makes up the data dictionary data base. Some of the possible attributes of entries in a data dictionary data base are listed below.

* Attribute name and synonyms
* Authorization password(s) for retrieval, update, delete, etc.
* Data type and format
* Range of values that may be stored
* Units in which the entity is represented, e.g., feet, meters
* Name of other entities that may initialize, update, or delete the data value
* Programming language(s) for which it is written
* Status, i.e. if the entity is in development, testing, damaged or production status
* Text (any text or comments may be written)

Layered above the dependent or independent implementation of a data dictionary is its functional ability to interact with a DBMS. A data dictionary is considered active if programs and processes depend on it for their metadata. A passive data dictionary is usually embedded within another system and therefore dependent on that system. An active data dictionary can be embedded within another system or completely independent with interfaces to many different systems. Its primary functions include identifying, locating, controlling, reporting, and manipulating metadata. With a completely active dictionary, users interface with the DBMS only through the dictionary. The scope of the activity varies, and a fully active data dictionary does not yet exist.

B. COMMERCIAL DATA DICTIONARY SYSTEMS

Data dictionary systems have evolved recently and become an increasingly useful tool for Data Base Administrators, auditors, systems analysts, programmers and users. In many organizations where data bases and information systems achieve a high degree of evolution and sophistication, the data dictionary becomes a practical necessity. NPS has a large number of potential data intensive applications well suited for a data base environment. It is plausible to assume that NPS could become increasingly dependent on a data base and information system. The data dictionary is a specialized data base management system, or application of an existing DBMS. The data base is a repository of descriptive information about the data bases, programs and other entities associated with information systems practice. There are two types of data dictionary systems, dependent and independent. An independent DD is self-contained and has its own reporting and maintenance programs. A dependent data dictionary is usually implemented as an application of a DBMS and is dependent on that particular DBMS to function.

1 . Independent

DBMS vendors are now marketing data dictionaries with growing capabilities for use specifically within their DBMS environments. However, some vendors who do not market a DBMS have developed data dictionaries to be nearly self-standing, that is, they do not require the use of a DBMS. At the same time, such products have interfaces to generate data descriptions specific to many popular DBMSs. The price range varies from about $15,000 to $40,000 depending on the particular vendor, version of the system and options included. Table 3.1 summarizes the interface capabilities of several of the major data dictionary systems. Data Catalog 2 and Datamanager are the only two listed that can be considered free-standing or independent. The choice is arbitrary and should not be interpreted as indicative of the author's preferences nor overall product ratings. Information provided in Table VIII was derived from [Ref. 8].

2. DBMS Dependent

A dependent data dictionary is more often than not a commercially available package, designed for a specific

TABLE VIII

COMMERCIAL DATA DICTIONARIES

SUPPLIER	NAME	DBMS REQ.	DBMS USED
CULLINET	IDD (INTEGRATED DATA DICTIONARY)	IDMS	IDMS
IBM CORP.	DB/DC DATA DICTIONARY SYSTEMS	IMS or DL/1	IMS
MANAGEMENT SYSTEMS AND PROGRAMMING LTD.	DATAMANAGER	NONE	IDMS TOTAL IMS ADABAS SYSTEM 2000
TSI INTERNATIONAL	DATA CATALOG 2	NONE	IMS TOTAL ADABAS DMS-1100 DATA MANAGER
UNIVAC	DATA DICTIONARY(DDS)	DMS-1100	DMS-1100

DBMS. The FOCUS Data Dictionary (FOCUSDD) is itself a FOCUS data base and is considered DBMS dependent. To an organization that only uses one DBMS, a dependent data dictionary can be an advantage. For example, FOCUSDD can use all FOCUS capabilities directly to analyze its contents and perform ad hoc analysis. Users need only learn one system versus two with an independent data dictionary.

The FOCUS Data Dictionary (FOCUSDD) is a comprehensive data management tool that can monitor, control, and audit applications. It serves as a central repository for the information on all elements of FOCUS systems. The following highlights were summarized from [Ref. 9:pp. 1-7].

* Menu-driven operation:

  FOCUSDD is an online, menu driven system. It is built around a MAIN MENU listing four basic options: Information Processing, Basic Reporting, System Maintenance, and entrance into FOCUS for ad hoc analysis. The first three options on the MAIN MENU lead to a sub-menu, which in turn breaks out into detailed menus providing a full range of options.

* Analysis of Master File and FOCEXEC information:

  The FOCUS Data Dictionary analyzes FOCUS Master Files, FOCEXECs, COBOL programs, and CMS EXECs. It records information for fields, files, input/output data sets and program CALLs used. After analyzing a Master File, it posts information regarding Masters, segments, cross-referenced segments and data fields into the dictionary. For FOCEXECs and other supported procedures, it generates cross-referenced reports listing referenced fields, system commands, and input/output data sets used. Analysis can be done on an individual file or group of files.

* Resource accounting facilities:

  This feature captures and maintains system resource utilization statistics for FOCEXECs by program. As a FOCEXEC is executed, the following information is collected and produced:

  - Date of last execution

  - Total number of executions

  - USERID of last execution

  - TOTAL CPU seconds

- VIRTUAL CPU seconds

- Total STARTIO commands

- Total CLOCK TIME

* Automatic dictionary update feature:

An automatic Dictionary Update feature provides facilities for automatically posting all supported file types into the dictionary. Updating is based on mnemonic values stored in the data base under a standard naming convention provided by FOCUSDD. All files containing these values will automatically be posted, analyzed, and updated.

The user is prompted to determine whether information about Masters, FOCEXECs and other programs no longer in existence should be deleted or maintained.

Menu-driven procedures for full-screen system maintenance include:

- Program description query/update

- Field description update

- Program deletion

- Master File description update

- Master File deletion

* Program change log facilities:

Provides a built-in audit trail of program and system modifications. The user can create, query, update or delete entries to the Log. This allows easy documentation and monitoring of all changes.

* Automatic generation of program documentation:

For each program in the dictionary, a formatted report is generated which displays program narrative and an input/output list including data bases accessed, external routines, and the actual source listing.

* Comprehensive set of standard reports:

A built-in series of sixteen standard reports display all available information on programs, data and their relationships. Report selection is menu-driven, featuring terminal or printed report generation. Key reports include:

- Catalogues of Master Files/Programs
- FOCEXEC listing by filename
- Keyword string query by FOCEXEC
- Program calls/called by program
- Fieldname string query by filename
- Field listing by FOCEXEC
- Resource analysis by USERID and program

FOCUSDD is completely menu-driven. It provides facilities for information storage, analysis, reporting, documentation and auditing. It can perform Software Nesting Analysis, resolve calling sequences up to ten levels deep, and provide a complete "calls/called by" analysis for user specified programs.

The DBA is provided a powerful means of controlling applications with FOCUSDD's Automatic Dictionary Update facility. Using standard naming conventions, this feature automatically posts all supported file types in the dictionary.

C. BENEFITS

The major benefits of a data dictionary derive from its flexibility to accommodate changes and its centralized location. To successfully implement any DDS, a commitment to the design, implementation, and proper usage is necessary from all levels of management. Programmers must restrict their work area and ensure that only valid and authorized changes are made to the system. Lastly, it is essential that a DBA be assigned to ensure compliance with the rules of the system. [Ref. 10:pp. 38-44]

Both types of data dictionaries have relative advantages. An independent DD can ensure consistency of data definitions by verifying and editing all data entries before storage. A dependent dictionary can be integrated into an existing DBMS environment with minimum user disruption. Also, a dependent dictionary can be useful in the case where all data is stored under a single DBMS.

There is no question that the stand-alone DD is potentially more powerful than the dependent. In some situations, such as multiple DBMSs, it may be the only way to maintain centralized control of data. Considering cost and performance, the conclusion to be drawn is that the dependent dictionary is more appropriate for organizations with existing data organized around a single DBMS. The independent dictionary is best suited for system start-up and multiple DBMS organizations.

V. ANALYSIS AND GENERAL DESIGN

A. FOCUS DESIGN CONSIDERATIONS

At the Naval Postgraduate School, FOCUS DBMS is used as the administrative data base facility. The Office of the Director of Admissions employs two programmer/analysts to develop application programs and maintain data integrity. Due to the large investment in time and money associated with FOCUS, the Director of Admissions was predisposed to its use for new applications. A formal requirements analysis was not conducted, but a recommendation to purchase PC FOCUS was made. When used on compatible micro-computer hardware, PC FOCUS would reduce the amount of mainframe CPU time required. Data could be downloaded to a personal computer in the Director of Admissions office for manipulation and reporting. Additionally, application programs could be generated on the PC and executed on the mainframe at times other than peak load.

FOCUS is based on the hierarchical model in which a parent segment may have one or many descendant segments. The child is limited to a single parent. This one-to-many relationship is denoted in diagrams by either projected boxes or a double headed arrow connecting parent to child. Solid lines are used to infer structural relationships and dotted lines for cross-reference or index to identical fields in other segments.

The discussion that follows is based on information contained in [Ref. 11].

1. Data Description Language

The description of a FOCUS file is typed into a CMS file whose filetype is MASTER. The CMS filename becomes the name by which the file is known to FOCUS. For example, the FOCUS file "STUDENTS" must have a file description stored with a CMS filename of "STUDENTS MASTER". Three classes of attributes are used to describe a FOCUS file. They are file attributes, segment attributes, and field attributes. Figure 5.1 below gives an example of a single segment master description.

FILE DECLARATION	FILENAME=	,SUFFIX=		
SEG DECLARATION	SEGNAME=	,SEGTYPE=	,PARENT=	
DATA OR	FIELDNAME=	,ALIAS=	,FORMAT=	,$
FIELD	FIELDNAME=	,ALIAS=	,FORMAT=	,$
DECLARATIONS	FIELDNAME=	,ALIAS=	,FORMAT=	,$
	FIELDNAME=	,ALIAS=	,FORMAT=	,$

Figure 5.1 Sample Focus Master Description

Attributes describing the file, segments, and fields of a FOCUS file are typed in free form comma delimited format. A dollar sign ($) signifies the end of the description of each element and a checking procedure can be used to locate typing errors and rule violations before to data entry. After data entry, it is no longer possible to make arbitrary changes to the file description. Any change to the master file that necessitates a change to the physical data base requires reorganization of the data. This does not imply weak logical/physical independence. Data must be reconstructed into a form that coincides with the master description. It would be nice if FOCUS would reindex files automatically. Nevertheless, the REBUILD utility provides options for rebuilding a FOCUS file, reorganizing a FOCUS file, and for indexing fields in a FOCUS file after data entry.

Both static and dynamic cross-referencing of files are available with advantages and disadvantages to each. Static cross references are specified in the master description and are therefore always active, but require the use of the REBUILD utility to change. Dynamic cross-reference is accomplished through the JOIN command. It does not take up file space by being pre-positioned in the master description and can be easily invoked when needed to link an entire file structure. On the negative side, the JOIN command must be issued during each session, can only link entire file structures, and is slower after first usage than a static cross reference.

Prior to use of static or dynamic cross-referencing, at least one field in the desired segment of each file must be of type "I" or indexed. This is accomplished through field attributes in the MASTER file. Any number of fields may be indexed on a segment, although only those which have common values in other files are practical candidates. The presence of the index is crucial for the operation of the cross reference facilities. Any number of external sources may locate and thereby share a segment because of it.

2. Data Manipulation Language

Once a MASTER file has been constructed and found free of errors, data can be entered into a FOCUS data base file. A non-procedural, fourth generation data manipulation language is used for all phases of data entry, manipulation, and reporting. The language is divided into two functional areas called the Transaction Processor and the Dialogue Manager. The purpose of the transaction processor language is to facilitate modification of information in a FOCUS data base without having to write a computer program. The transaction processor is entered by typing the FOCUS command MODIFY followed by the name of the file to be changed.

The transaction processor has the ability to accept input in several forms. FIXFORM specifies fixed format data with each field appearing in a set position of the record. When FIXFORM is used the name of the data fields and the number of characters to be processed must be specified. Under normal usage FIXFORM does not specify the source of the data input transactions. Instead, the subcommand, DATA ON, is usually placed at the end of the procedure. The PROMPT subcommand specifies that the user will enter data interactively in response to prompts from FOCUS. When PROMPT is used, FOCUS will request that the operator type the data values in response to prompting messages. Only one field at a time is requested, but the operator has the option of using free form comma delimited format to input several values at a time. FOCUS will then skip ahead and resume prompting for any values not yet entered. Preformatted CRT screens for fill-in-the-blank data entry can be designed using the CRTFORM subcommand. The FREEFORM subcommand can be used to alter the natural order of data input from that described in the MASTER file.

The basic operation during transaction processing is to match values from a transaction to corresponding values in a data base and take action depending on the status of the match. The MATCH subcommand is used to designate fields to match. All key fields in each segment must be specified. Fieldnames can always be referenced by either their full fieldname, alias, or shortest unique truncation. The actions to be taken following the MATCH subcommand are specified through the ON MATCH and ON NOMATCH subcommands. A short description of each of the possible actions is given below.

REJECT	The transaction is considered a duplicate.
UPDATE	If a match is found, the fields specified will be updated.
DELETE	The matched record segment, all dependent records, references, and indexes are deleted.
COMPUTE	The expressions that follow may refer to transaction values or data base values before or at the point of match. These new values may be used to update the data base.
INCLUDE	A data base record is to be included. This is the basic input process.
VALIDATE	The expressions may refer to the same values as COMPUTE. If the logic of the expression evaluates to false, the transaction is rejected.
TYPE	A message may be typed in response to MATCH or NOMATCH logic.
PROMPT	Prompts user for specified fields.
FREEFORM	Additional data is read from a transaction file.
CRTFORM	Beginning or continuation of fill-in-the-blank CRT screen.
FIXFORM	Same as FREEFORM
CONTINUE	This is the default option when a further MATCH subcommand in present, but it is recommended that action be defined explicitly.
GOTO	Unconditional branch to named CASE.

IF-GOTO   Conditional branch to named CASE.

It is possible to store FOCUS transaction processing commands in a file for repetitive use or execution at a later time. Any CMS filename and filetype can be used to store the procedure. If only a filename is provided, a filetype of FOCEXEC is assumed .

With the Dialogue Manager, a stored procedure may contain variable information for which a value is provided only at the time of execution. The variables can be used to represent data as numeric constants, dates, or to conduct a dialogue by prompting the operator for a response. The first character of the variable must be an ampersand.

The Dialogue Manager is invoked by typing the FOCUS command EXEC followed by the name of the procedure. Together, the Dialogue Manager and DDL constitute a semi-structured programming language. Any line containing Dialogue Manager commands or GOTO labels must begin with a dash. The EXEC command can be embedded in a program to call another module. On completion of the called module, the Dialogue Manager returns to the next step in the calling program.

The sequential execution of one Dialogue Manager statement after another can be altered by use of branching statements. The GOTO command can be used separately, in an unstructured mode, to branch to a label specified elsewhere in the program. This can result in complicated coding. To eliminate this problem, a structured programming discipline must be maintained. IF-GOTO logic and labels should be used only for sequential or closed loop execution. Branching is most often used to test the values of prompted variables and then select the procedure to be executed.

Several other commands and labels are frequently used. Messages can be typed from a stored procedure on Dialogue Manager lines beginning with the label -TYPE. Variables may be embedded in the text of the line. -EXIT, -QUIT, and -RUN are three labels used to control the execution and return characteristics of a FOCUS stored procedure. Individual lines of a stored procedure are stacked, awaiting execution, until either the label -EXIT or -Run is encountered. An implied -EXIT exists after the last line in a procedure. When either an implicit or explicit -EXIT is encountered processing of lines in the procedure is ended and execution of the stacked lines begins. Control is returned to the next higher program level or native FOCUS. With -QUIT, the return characteristics are the same but the stacked lines are not executed. The -RUN label exits the procedure and executes the stacked lines. Processing resumes at the line following -RUN. Table IX summarizes the effects of the three labels.

TABLE IX

FOCUS DIALOGUE MANAGER LABELS

LABEL	EXECUTES STORED LINES	RETURNS TO
-EXIT	YES	Next higher program level or native FOCUS
-QUIT	NO	Next higher program level or native FOCUS
-RUN	YES	Line following -RUN

If a stored module, called by another procedure, does not contain an -EXIT label or an END command, the terminal will remain open for interactive processing. When processing or queries are complete, the user enters QUIT to return to the calling procedure. This technique is used in the design of the data dictionary discussed later in this chapter.

B. DATA DICTIONARY

During the design phase of any modular application, standard interfaces are specified. This can be accomplished through communication between programmers to identify parameter passing requirements such as global variables. Verbal communication is often time consuming, error prone, and confusing. Another method for standardization and control is the data dictionary. Using this method, the DBA can specify what data elements and relationships are available for programmer's use. The use of a data dictionary during the program design phase is one specific application of standardization and sharing.

In some circles, a data dictionary is considered to have only limited benefits during the development of new data processing systems. The real benefit is in the reduction of maintenance costs [Ref. 7:p. 90]. The dividing line between development and maintenance is arbitrary. It is sometimes hard to say when development ends and maintenance begins. An explanation for this opinion is, when compared to the vast benefits gained during the remainder of the system life cycle, development is only a small percentage of the effort. Whatever the explanation, the judgment is in error: "A data dictionary can be of particular value to systems analysts/designers in the three phases of system development: (1) analysis, (2) design, and (3) implementation." [Ref. 10:p. 105] It is acknowledged in Government and private industry that an increase in productive time and money expended during software development can significantly reduce total life cycle costs.

A data dictionary can be an invaluable aid during applications program development. However, not all functions of the FOCUS DATA DICTIONARY, described in Chapter 3, are required or even useful during the development stage.

The most beneficial dictionary functions to a programmer are listed below.

* reports which display all available information on programs, data, and their relationships

* automatic dictionary update for new or changed Masters and FOCEXECs.

Reports which display information on data and their relationships can yield many benefits. This one function can reduce data redundancy and develop standards by providing the system designer with current data descriptions and naming conventions.

It is possible, with the FOCUS 4GL, to implement these useful functions without the expense of a commercial data dictionary package. Nevertheless, a basic system development dictionary would not be of much use during later life cycle stages.

1. Planning

It is necessary to plan the scope and objectives of a data dictionary before construction can begin. Here, the purpose of the dictionary is to aid in the development of application programs. The primary objective is the interactive cross-referencing, verifying, and updating of data about data. A programmer who is able to determine existing relationships and names from a data dictionary will be able to make use of those relationships and standard naming conventions.

The proposed system is narrow in scope. It is designed as a FOCUS DBMS dependent application. Features include ad hoc query capability, standard reports on metadata, and dictionary maintenance.

Although automatic dictionary updates are useful to the programmer, they are considered non-essential in this case. A description of how to implement an automatic update feature is given in the design phase.

2. Requirements Definition

The outputs from the design data dictionary include the following.

* FOCUS file definitions/descriptions
* Segment definitions/descriptions
* Field description/alias
* FOCUS file summary report
* FOCUS EXEC descriptions
* Variable names/descriptions
* Called/Called by analysis

The FOCUS file, segment, and field descriptions and summary report are used to determine where and how data is stored. All the information contained in these reports, with the exception of the narrative description, can be gleaned from the FOCUS Master Description.

The file description contains the file name, a narrative description, file type, and number of segments. In addition, the individual or organization responsible for maintenance on the file is listed. The same type information is contained in the segment and field reports.

Details on FOCUS EXECs are contained in variable, FOCUS EXEC, and called/called by reports. The FOCUS EXEC report is a quick reference to determine the purpose of named EXECs. A simple naming convention is imposed through compliance with a standard prefix-postfix routine. The prefix identifies EXEC purpose and the postfix specifies a type. For example, if the purpose of an EXEC were to add file information, it would be named ADDFILE. If the purpose were to delete EXEC information, it would be named DELEXEC. The called/called by analysis can be used to determine the impact of a change to a specific EXEC.

The ability to add, change, or delete items in the dictionary is called dictionary maintenance. It is accomplished through a menu driven utility for either FOCUS files or EXECs.

Ad hoc queries can be made two ways. First, by exiting the dictionary and returning to native FOCUS, the DML can be used to make queries. The disadvantage is that knowledge of data structures is required to execute dynamic joins. The second method is to make the menu selection that best answers the query, then at the end of execution, make the ad hoc query prior to entering quit. This technique was discussed earlier in this chapter under the Dialogue Manager.

3. Design

Table X contains the data necessary to meet the requirements above. Two FOCUS Master Files, shown in Table XI, were defined using these data elements.

TABLE X

DICTIONARY DATA ELEMENTS

FOCUS EXECS	MASTER FILES
FOCUS EXEC name	Master File name
EXEC purpose	File type
Variable name	File description
Variable format	Segment name
Variable description	Segment parent
File used by EXEC	Segment type
EXEC called by EXEC	Segment description
	Field name
	Alias
	Field format
	Field type
	Field description
	key

TABLE XI

FOCUS DESIGN DICTIONARY MASTER FILES

FOCUS MASTER FILE EDICTION

```
FILENAME=EDICTION,SUFFIX=FOC
SEGNAME=EXECS,SEGTYPE=S1
    FIELDNAME=EXEC_NAME    ,ALIAS=EN  ,FORMAT=A8   ,$
    FIELDNAME=PURPOSE      ,ALIAS=DOES,FORMAT=A25   ,$
SEGNAME=VARIABLE,PARENT=EXECS,SEGTYPE=S1
    FIELDNAME=VAR_NAME     ,ALIAS=VN  ,FORMAT=A8    ,$
    FIELDNAME=VAR_FORMAT   ,ALIAS=VFMT,FORMAT=A5    ,$
    FIELDNAME=VAR_DESCRIPT,ALIAS=VDES,FORMAT=A25   ,$
SEGNAME=FILE_NAM,PARENT=EXECS,SEGTYPE=S1
    FIELDNAME=FILE_NAME    ,ALIAS=FLN ,FORMAT=A8   ,TYPE=I,$
SEGNAME=CALLED,PARENT=EXECS,SEGTYPE=S1
    FIELDNAME=CALLED_EXEC ,ALIAS=CE  ,FORMAT=A8    ,$
```

FOCUS MASTER FILE FDICTION

```
FILENAME=FDICTION,SUFFIX=FOC
SEGNAME=FILES,SEGTYPE=S1
    FIELDNAME=FILE_NAME    ,ALIAS=FLN ,FORMAT=A8   ,TYPE=I,$
    FIELDNAME=FILE_TYPE    ,ALIAS=FTYP,FORMAT=A3    ,$
    FIELDNAME=NUM_SEGS     ,ALIAS=NS  ,FORMAT=I1    ,$
    FIELDNAME=FILE_DESCRIP,ALIAS=FDES,FORMAT=A25   ,$
    FIELDNAME=MAINTAINE_BY,ALIAS=FMAN,FORMAT=A12   ,$
SEGNAME=SEGMENTS,PARENT=FILES,SEGTYPE=S1
    FIELDNAME=SEG_NAME     ,ALIAS=SEGN,FORMAT=A8    ,$
    FIELDNAME=CHILD_OF     ,ALIAS=SPAR,FORMAT=A8    ,$
    FIELDNAME=SEG_TYPE     ,ALIAS=STYP,FORMAT=A3    ,$
    FIELDNAME=SEG_DESCRIPT,ALIAS=SDES,FORMAT=A25   ,$
SEGNAME=FIELDS,PARENT=SEGMENTS,SEGTYPE=S1
    FIELDNAME=FIELD_NAME   ,ALIAS=FDN ,FORMAT=A12   ,$
    FIELDNAME=ALTERNATE    ,ALIAS=ALT ,FORMAT=A4    ,$
    FIELDNAME=FIELD_FORMAT,ALIAS=FFMT,FORMAT=A5    ,$
    FIELDNAME=FIELD_TYPE   ,ALIAS=FDTP,FORMAT=A1    ,$
    FIELDNAME=KEY          ,ALIAS=    ,FORMAT=A1    ,$
    FIELDNAME=FIELD_DESCRI,ALIAS=FDDE,FORMAT=A25   ,$
```

Figures 5.2 and 5.3 depict the logical FOCUS structures derived from the Master Files, which comprise the data storage structures for the dictionary. FOCUS source code for the data dictionary and sample reports are contained in Appendix A.

FOCUS has the ability to read files other than those it creates itself. A comma delimited file is one in which individual fields are separated by a comma. A FOCUS Master

```
+-----------------------------------------------------------------+
|                                                                 |
|          EXECS                                                  |
|  01       S1                                                    |
| **************                                                  |
| *EXEC_NAME   **                                                 |
| *PURPOSE     **                                                 |
| *            **                                                 |
| *            **                                                 |
| *            **                                                 |
| ***************                                                 |
|  **************                                                 |
|       I                                                         |
|       +-----------------+-----------------+                     |
|       I                 I                 I                     |
|       I VARIABLE        I FILE_NAM        I CALLED              |
|  02   I S1       03     I S1       04     I S1                  |
| **************    **************    **************             |
| *VAR_NAME    **   *FILE_NAME   **I  *CALLED_EXEC **            |
| *VAR_FORMAT  **   *            **   *             **            |
| *VAR_DESCRIPT**   *            **   *             **            |
| *            **   *            **   *             **            |
| *            **   *            **   *             **            |
| ***************   ***************   ***************            |
|  **************    **************    **************            |
|                                                                 |
|                                                                 |
+-----------------------------------------------------------------+
```

Figure 5.2 Structure of Focus File Ediction

```
         FILES
 01      S1
**************
*FILE_NAME   **I
*FILE_TYPE   **
*NUM_SEGS    **
*FILE_DESCRIP**
*            **
***************
**************
       I
       I
       I
       I SEGMENTS
 02    I S1
**************
*SEG_NAME    **
*CHILD_OF    **
*SEG_TYPE    **
*SEG_DESCRIPT**
*            **
***************
 **************
       I
       I
       I
       I FIELDS
 03    I S1
**************
*FIELD_NAME  **
*ALTERNATE   **
*FIELD_FORMAT**
*FIELD_TYPE  **
*            **
***************
 **************
```

Figure 5.3   Structure of Focus File Fdiction

File is designed as a comma delimited file and as such can be read directly by FOCUS.

The design data dictionary in Appendix A can be made partially active by taking advantage of FOCUS Master Files comma delimited form. If the dictionary Master File were assigned a file type of external, data from individual Master Files already in existence could be made available to the user. This would alleviate the need to enter information into the dictionary which is already contained in a Master File. FOCUS also makes provisions for a narrative description within a Master File.

Further attempt to activate the design dictionary is not recommended. The inhouse development of an active data dictionary can be a costly and time consuming process. The Director of Admissions Office does not have the assets available to accomplish that. It is recommended that a Data Base Administrator be assigned from the Director of Admissions Office and that the FOCUS Data Dictionary be purchased and implemented.

C. TRANSCRIPT SUMMARY APPLICATION

The planning and requirements phase of the transcript summary application were covered in Chapter 2. Table XII lists the FOCUS Master File descriptions. The resulting logical structures, shown in Figures 5.4, 5.5, and 5.6, were dictated by the characteristics of the external files supplied by the Naval Academy.

TABLE XII

TRANSCRIPT SUMMARY APPLICATION MASTER FILES

FOCUS MASTER FILE ADMISSIO

```
FILENAME=ADMISSIO,SUFFIX=FOC
SEGNAME=STUDENT,SEGTYPE=S1
    FIELDNAME=STUD_ID      ,ALIAS=SID ,FORMAT=A7    ,$
    FIELDNAME=STUD_NAME    ,ALIAS=SN  ,FORMAT=A17   ,$
    FIELDNAME=SOC_SEC_NUM  ,ALIAS=SSN ,FORMAT=A10   ,$
    FIELDNAME=MAJOR        ,ALIAS=MAJ ,FORMAT=A4    ,$
    FIELDNAME=SEX          ,ALIAS=SEX ,FORMAT=A2    ,$
    FIELDNAME=NATIONALITY  ,ALIAS=RACE,FORMAT=A8    ,$
SEGNAME=COURSE,SEGTYPE=S1
    FIELDNAME=COURSE_ID    ,ALIAS=CID ,FORMAT=A6    ,$
    FIELDNAME=LETTER_GRADE,ALIAS=LG   ,FORMAT=A2    ,$
    FIELDNAME=SEMESTER     ,ALIAS=WHEN,FORMAT=A3    ,$
```

FOCUS MASTER FILE AVAIL

```
FILENAME=AVAIL,SUFFIX=FOC
SEGNAME=DESCRIPT,SEGTYPE=S1
    FIELDNAME=COURSE_ID    ,ALIAS=CID ,FORMAT=A6   ,TYPE=I,$
    FIELDNAME=CREDIT_HOURS,ALIAS=CHRS,FORMAT=I1     ,$
```

FOCUS MASTER FILE REQ

```
FILENAME=REQ,SUFFIX=FOC
SEGNAME=DESCRIPT,SEGTYPE=S1
    FIELDNAME=COURSE_ID    ,ALIAS-CID ,FORMAT=A6   ,TYPE=I,$
    FIELDNAME=SUBJ_AREA    ,ALIAS=SA  ,FORMAT=A7    ,$
```

The source code and sample output for a prototype application that summarizes transcript information are given in Appendix B. The prototype can be modified to read data from the fixed format external file described in Chapter 2.

```
          STUDENT
  01      S1
**************
*STUD_ID     **
*STUD_NAME   **
*SOC_SEC_NUM **
*MAJOR       **
*            **
***************
 **************
       I
       I
       I
       I COURSE
 02    I S1
**************
*COURSE_ID   **
*LETTER_GRADE**
*SEMESTER    **
*            **
*            **
***************
 **************
```

Figure 5.4   Structure of Focus File Admissio

The external file contains a group of adjacent fields that are repeated in the same record. (ie. Course information is repeated numerous times for each student.) Such a structure can be described to FOCUS by assigning the non-repeating portion to one segment, and the repeating group to another, which is its descendent. By assigning a Segment attribute of VARIABLE to the repeating group, FOCUS is informed that the length of the physical external file varies. The number

```
+----------------------------------------------------------+
|                                                          |
|                     DESCRIPT                             |
|             01      S1                                   |
|            **************                                |
|            *COURSE_ID   **I                              |
|            *SUBJ_AREA   **                               |
|            *            **                               |
|            *            **                               |
|            *            **                               |
|            ***************                               |
|             **************                               |
|                                                          |
|                                                          |
+----------------------------------------------------------+
```

Figure 5.5   Structure of Focus File Req

```
+----------------------------------------------------------+
|                                                          |
|                     DESCRIPT                             |
|             01      S1                                   |
|            **************                                |
|            *COURSE_ID   **I                              |
|            *CREDIT_HOURS**                               |
|            *            **                               |
|            *            **                               |
|            *            **                               |
|            ***************                               |
|             **************                               |
|                                                          |
|                                                          |
|                                                          |
+----------------------------------------------------------+
```

Figure 5.6   Structure of Focus File Avail

of occurrences of the repeating group is then determined for each record by dividing the number of characters read by the length of the repeating segment. Since the prototype was designed using the same logical structure as that of the external file, this modification would not necessitate changes to program logic.

VI. CONCLUSIONS

A. IDENTIFICATION OF NEED

Because of the size of the programming staff in the Office of the Director of Admissions and the limited scope of present FOCUS application programs, it might be concluded that a data dictionary is not required. Nevertheless, now is the time to implement a DDS. The Administration has already embraced the DBMS approach and the programming backlog is increasing. The longer implementation is delayed, the greater the chance that data redundancy and the loss of data integrity will erode system credibility. In addition, programmer response time to new applications may increase because of the maintenance effort required for existing applications.

One major benefit of a 4GL DBMS like FOCUS is its ability to narrow the gap between users and programmers. As users become more familiar with the system, they should be able to develop application programs of their own. This should result in a reduction of the programming backlog. On the other hand, with this new found familiarity with the system, management might envision and request more program development. A properly implemented and maintained data dictionary can aid in programmer effectiveness and provide management with more effective control over data.

B. BENEFITS

Even the most basic form of a data dictionary, when properly implemented, can be of great use during the program development cycle. Conversely, poor design, implementation, or use will make problems like data integrity and redundancy even worse.

The major benefits derived from use of the data dictionary (Appendix A) during the design phase of the transcript summary application (Appendix B) are listed below:

* Reduced data redundancy

* Enhanced data integrity

* Documentation of existing relationships

* Simplified system maintenance

* Highlighted standard naming conventions

* Provided data element reference

Benefits derived from a data dictionary are proportional to its size. Since the transcript summary program spanned only three FOCUS files, it is hard to say that the data dictionary provided any great advantage. Nevertheless, access to data elements and their logical structure enhanced the programmer's ability to locate and reduce data redundancy.

Implementation of a data dictionary requires the standardization of data items. For a large existing system in an organization with multiple departments, this can be complex. Even in the simple application discussed here, the standardization of data items greatly contributes to long range data integrity.

During the design of the transcript application, information on which programs use the same data type and how they relate was required. This is in essence a limited view of the logical data structure and relationships. It is required for the proper access of required data, using dynamic joining when necessary.

Maintenance can be defined many ways. Regardless of whether it is considered as modifications after delivery or after the first successful execution, the benefits of the data dictionary are the same. If the dictionary is updated along with program modifications, it becomes consistent, reliable documentation that is essential for program maintenance.

At a basic level, a data dictionary is composed of data elements and their description. This reference list can be used to evaluate the impact of proposed changes, prior to their occurrence. By its nature it provides a way to highlight standard naming conventions. New data elements to be entered in the dictionary must be consistent with existing naming conventions.

C. SYNOPSIS

This thesis described the design, implementation, and use of a basic data dictionary. The design and development of an undergraduate transcript summary application for the Director of Admissions at NPS was used to evaluate the benefits of the data dictionary. The concepts of dependent and independent dictionaries were discussed and the fundamental principles were applied to the dictionary design. A comparison of three data base models and a summary of commercial data base management systems and data dictionaries was made. The advantages and disadvantages in the development and use of 4GLs were discussed.

As the scope of data base applications in an organization grows, the tendency is to define new data elements and structures rather than interpret the data that already exists. The data dictionary is a tool that provides programmers with access to standard data items and structures. It is the responsibility of the DBA to ensure that these standards are maintained. Management, users, and data processing personnel must all be committed to this standardization concept before any benefits can be realized.

LIST OF REFERENCES

1. Kroenke, David, Database Processing, Science Research Associates, Inc., 1983.

2. Martin, James, Computer Data-Base Organization, Prentice-Hall, Inc., 1975.

3 Codd, E. F., "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM, No. 6, June 1970.

4. Cobb, Richard H., "In Praise of 4GLS," Datamation, v. 31, July 1985.

5. Grant, F. J., "The Downside of 4GLS," Datamation, v. 31, July 1985.

6. Patel, Has, "How To Upgrade A DBMS," Datamation, September 1981.

7. Durell, William, "Disorder to Dicipline Via the Data Dictionary," Journal Of Systems Management, May 1983.

8. International Data Base Management System Symposium, The 1984 Update to Data Base Management Systems, 31 August 1983.

9. Information Builders, Inc., Application Product Description, FOCUS Data Dictionary, July 1983.

10. Duyn, J. Van, Developing a Data Dictionary System, Prentice-Hall, Inc., 1982.

11. Information Builders, Inc., FOCUS Users Manual, July 1984.

APPENDIX A

DATA DICTIONARY SOURCE CODE

```
*********************************************************************
*                      MODULE: MAINMENU FOCEXEC                     *
*                      WRITTEN BY: BOB REPP                         *
*                      CALLED BY: PROJMENU FOCEXEC                  *
*                      CALLS: MANTMENU,FILEMENU,EXECMENU FOCEXEC    *
*                      PURPOSE: MAIN MENU FOR THE DATA              *
*                        DICTIONARY                                 *
*                                                                   *
*********************************************************************
SET MSG=OFF
-CRTCLEAR
-BEGIN
-CRTCLEAR
-TYPE THIS IS THE MAIN MENU FOR THE DATA DICTIONARY
-TYPE
-TYPE
-TYPE
-TYPE
-TYPE
-TYPE
-TYPE  |---MAIN MENU-----------------|
-TYPE  |                              ---------------------------------|
-TYPE  |                                                               |
-TYPE  |      INFORMATION ON FOCUS FILES ...... <F>                    |
-TYPE  |      INFORMATION ON FOCEXECS ......... <E>                    |
-TYPE  |      DICTIONARY MAINTENANCE .......... <M>                    |
-TYPE  |      QUIT ............................ <Q>                    |
-TYPE  |                                                               |
-TYPE  |---------------------------------------------------------------|
-TYPE
-TYPE
-TYPE
-PROMPT &SELECT. WHAT IS YOUR CHOICE?.
-IF         &SELECT IS 'F' GOTO FILEMENU
-   ELSE IF &SELECT IS 'E' GOTO EXECMENU
-   ELSE IF &SELECT IS 'M' GOTO MAINT
-   ELSE IF &SELECT IS 'Q' GOTO EXIT
-   ELSE GOTO BEGIN;
-FILEMENU
EXEC FILEMENU
-RUN
-CRTCLEAR
-GOTO BEGIN
-EXECMENU
EXEC EXECMENU
-RUN
-CRTCLEAR
-GOTO BEGIN
-MAINT
EXEC MANTMENU
-RUN
-CRTCLEAR
-GOTO BEGIN
-EXIT
```

```
**********************************************************************
*                      MODULE: MANTMENU FOCEXEC                       *
*                      WRITTEN BY: BOB REPP                           *
*                      CALLED BY: MAINMENU FOCEXEC                    *
*                      CALLS: MODIFILE FOCEXEC, MODIEXEC FOCEXEC      *
*                      PURPOSE: DICTIONARY MAINTENANCE MENU           *
*                                                                     *
*                                                                     *
**********************************************************************
-CRTCLEAR
-FIRST
-TYPE THIS IS THE MAINTENANCE MENU FOR THE DATA DICTIONARY
-TYPE
-TYPE
-TYPE
-TYPE
-TYPE  ¦---MAIN MENU----------------¦______________________________
-TYPE  ¦                                                           ¦
-TYPE  ¦ ¦---MAINTENANCE MENU---------¦______________________________
-TYPE  ¦ ¦                                                           ¦
-TYPE  ¦ ¦                                                           ¦
-TYPE  ¦ ¦             MODIFY FILE DATA .......... <1>               ¦
-TYPE  ¦ ¦             MODIFY EXEC DATA .......... <2>               ¦
-TYPE  ¦ ¦             EXIT (to main menu) ....... <X>               ¦
-TYPE  ¦_¦                                                           ¦
-TYPE   -¦                                                           ¦
-TYPE    ¦___________________________________________________________¦
-TYPE
-TYPE
-TYPE
-PROMPT &WHICH. WHAT IS YOUR CHOICE?.
-IF          &WHICH EQ 1 GOTO MODIFILE
-   ELSE IF &WHICH EQ 2 GOTO MODIEXEC
-   ELSE IF &WHICH IS 'X' GOTO EXIT
-   ELSE GOTO FIRST;
-MODIFILE
EXEC MODIFILE
-RUN
-CRTCLEAR
-GOTO FIRST
-MODIEXEC
EXEC MODIEXEC
-RUN
-CRTCLEAR
-GOTO FIRST
-EXIT
EOF:
```

```
*********************************************************************
*                      MODULE: MODIEXEC FOCEXEC                     *
*                      WRITTEN BY: BOB REPP                         *
*                      CALLED BY: MANTMENU FOCEXEC                  *
*                      CALLS: ADDEXEC, CHGEXEC, DELEXEC FOCEXEC     *
*                      PURPOSE: EXEC MAINTENANCE MENU               *
*                                                                   *
*                                                                   *
*********************************************************************
-CRTCLEAR
-PRIME
-TYPE THIS IS THE EXEC MAINTENANCE MENU FOR THE DATA DICTIONARY
-TYPE
-TYPE
-TYPE
-TYPE
-TYPE  |---MAIN MENU----------------|______________________________
-TYPE  |                                                           |
-TYPE  | |---EXEC MAINTENANCE MENU----|_______________________________
-TYPE  | |                                                           |
-TYPE  | |                                                           |
-TYPE  | |                ADD EXEC INFORMATION ...... <1>            |
-TYPE  | |                UPDATE EXEC INFORMATION ... <2>            |
-TYPE  | |                DELETE EXEC INFORMATION ... <3>            |
-TYPE  |_|                EXIT (to main menu) ....... <X>            |
-TYPE   -|                                                           |
-TYPE    |                                                           |
-TYPE    |___________________________________________________________|
-TYPE
-TYPE
-TYPE
-PROMPT &WHAT. WHAT IS YOUR ANSWER?.
-IF        &WHAT EQ  1  GOTO ADDEXEC
-  ELSE IF &WHAT EQ  2  GOTO CHGEXEC
-  ELSE IF &WHAT EQ  3  GOTO DELEXEC
-  ELSE IF &WHAT IS 'X' GOTO EXIT
-  ELSE GOTO PRIME;
-ADDEXEC
EXEC ADDEXEC
-RUN
-CRTCLEAR
-GOTO PRIME
-CHGEXEC
EXEC CHGEXEC
-RUN
-CRTCLEAR
-GOTO PRIME
-DELEXEC
EXEC DELEXEC
-RUN
-CRTCLEAR
-GOTO PRIME
-EXIT


*********************************************************************
*                      MODULE: ADDEXEC FOCEXEC                      *
*                      WRITTEN BY: BOB REPP                         *
*                      CALLED BY: MODIEXEC FOCEXEC                  *
*                      CALLS: ADD1EXEC,ADD2EXEC,ADD3EXEC,ADD4EXEC   *
*                      PURPOSE: ADD MENU UNDER DICTIONARY MAINTENANCE*
*                                                                   *
*                                                                   *
*********************************************************************
-FIRST
-CRTCLEAR
-TYPE THIS IS THE ADD MENU UNDER DICTIONARY MAINTENANCE
-TYPE
-TYPE
-TYPE
```

```
-TYPE
-TYPE |  ‾‾MAIN‾MENU‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾T________________________________
-TYPE |                                                                    |
-TYPE | |‾‾‾MAINTENANCE‾MENU‾‾‾‾‾‾‾‾‾‾|________________________________
-TYPE | |                                                                    |
-TYPE | | |‾‾‾ADD‾EXEC‾INFORMATION‾‾‾‾‾|________________________________
-TYPE | | |                                                                    |
-TYPE | | |       ADD NEW EXECS ................................ <1>           |
-TYPE | | |       ADD VARIABLES TO EXISTING EXEC ............. <2>           |
-TYPE |_| |       ADD MASTER FILES USED BY AN EXISTING EXEC .. <3>           |
-TYPE  ‾| |       ADD EXECS CALLED BY AN EXISTING EXEC ....... <4>           |
-TYPE   |_|       EXIT ....................................... <X>           |
-TYPE    ‾|                                                                    |
-TYPE     |                                                                    |
-TYPE     |--------------------------------------------------------------------|
-TYPE
-PROMPT &CHOOSE. WHAT IS YOUR CHOICE?.
-IF         &CHOOSE EQ 1 GOTO ADD1EXEC
-   ELSE IF &CHOOSE EQ 2 GOTO ADD2EXEC
-   ELSE IF &CHOOSE EQ 3 GOTO ADD3EXEC
-   ELSE IF &CHOOSE EQ 4 GOTO ADD4EXEC
-   ELSE IF &CHOOSE IS 'X' GOTO EXIT
-   ELSE GOTO FIRST;
-ADD1EXEC
EXEC ADD1EXEC
-RUN
-CRTCLEAR
-GOTO FIRST
-ADD2EXEC
EXEC ADD2EXEC
-RUN
-CRTCLEAR
-GOTO FIRST
-ADD3EXEC
EXEC ADD3EXEC
-RUN
-CRTCLEAR
-GOTO FIRST
-ADD4EXEC
EXEC ADD4EXEC
-RUN
-CRTCLEAR
-GOTO FIRST
-EXIT


**********************************************************************
*                       MODULE: ADD1EXEC FOCEXEC                     *
*                       WRITTEN BY: BOB REPP                         *
*                       CALLED BY: ADDEXEC FOCEXEC                   *
*                       CALLS: EDICTION FOCUS, EDICTION MASTER       *
*                       PURPOSE: ADD NEW EXECS TO DATABASE           *
*                                                                    *
*                                                                    *
**********************************************************************
-CRTCLEAR
MODIFY FILE EDICTION
PROMPT EXEC NAME
MATCH EXEC NAME
     ON MATCH REJECT
     ON NOMATCH PROMPT PURPOSE
 NOMATCH INCLUDE
DATA
```

```
*********************************************************************
*                      MODULE: ADD2EXEC FOCEXEC                     *
*                      WRITTEN BY: BOB REPP                         *
*                      CALLED BY: ADDEXEC FOCEXEC                   *
*                      CALLS: EDICTION FOCUS, EDICTION MASTER       *
*                      PURPOSE: ADD VARIABLES TO THE DATA           *
*                        DICTIONARY FOR EXECS ALREADY IN THE        *
*                        DICTIONARY                                 *
*********************************************************************
-CRTCLEAR
MODIFY FILE EDICTION
PROMPT EXEC_NAME VAR_NAME
MATCH EXEC_NAME
     ON MATCH CONTINUE
     ON NOMATCH REJECT
MATCH VAR_NAME
     ON MATCH REJECT
     ON NOMATCH PROMPT VAR_FORMAT VAR_DESCRIPT
     ON NOMATCH INCLUDE
DATA


*********************************************************************
*                      MODULE: ADD3EXEC FOCEXEC                     *
*                      WRITTEN BY: BOB REPP                         *
*                      CALLED BY: ADDEXEC FOCEXEC                   *
*                      CALLS: EDICTION FOCUS, EDICTION MASTER       *
*                      PURPOSE: ADD MASTER FILES, USED BY AN        *
*                        EXEC, TO THE DATA DICTIONARY               *
*                                                                   *
*********************************************************************
-CRTCLEAR
MODIFY FILE EDICTION
PROMPT EXEC_NAME FILE_NAME
MATCH EXEC_NAME
     ON MATCH CONTINUE
     ON NOMATCH REJECT
MATCH FILE_NAME
     ON MATCH REJECT
     ON NOMATCH INCLUDE
DATA


*********************************************************************
*                      MODULE: ADD4EXEC FOCEXEC                     *
*                      WRITTEN BY: BOB REPP                         *
*                      CALLED BY: ADDEXEC FOCEXEC                   *
*                      CALLS: EDICTION FOCUS, EDICTION MASTER       *
*                      PURPOSE:   ADD EXECS TO THE DATA             *
*                        DICTIONARY THAT ARE CALLED BY AN EXEC      *
*                        ALREADY IN THE DICTIONARY                  *
*********************************************************************
-CRTCLEAR
MODIFY FILE EDICTION
PROMPT EXEC_NAME CALLED_EXEC
MATCH EXEC_NAME
     ON MATCH CONTINUE
     ON NOMATCH REJECT
MATCH CALLED_EXEC
     ON MATCH REJECT
     ON NOMATCH INCLUDE
DATA
```

```
*********************************************************************
*                          MODULE: CH6EXEC FOCEXEC                  *
*                          WRITTEN BY: BOB REPP                     *
*                          CALLED BY: MODIEXEC FOCEXEC              *
*                          CALLS: CH61EXEC, CH62EXEC FOCEXEC        *
*                          PURPOSE: UPDATE EXEC MENU UNDER          *
*                            DICTIONARY MAINTENANCE                 *
*                                                                   *
*********************************************************************
-MENU
-CRTCLEAR
-TYPE0 THIS IS THE UPDATE MENU UNDER DICTIONARY MAINTENANCE
-TYPE
-TYPE
-TYPE
-TYPE
-TYPE |  --MAIN MENU-----------------|_________________________________|
-TYPE |                                                                 |
-TYPE |  |---MAINTENANCE MENU---------|_________________________________|
-TYPE |  |                                                                |
-TYPE |  |  |---UPDATE EXEC INFORMATION--|________________________________|
-TYPE |  |  |                                                                 |
-TYPE |  |  |       UPDATE VARIABLE FORMAT OR DESCRIPTION ...... <1>         |
-TYPE |  |  |       UPDATE EXEC PURPOSE ........................ <2>         |
-TYPE |_ |  |       EXIT ....................................... <X>         |
-TYPE   -|  |                                                                 |
-TYPE    |_ |                                                                 |
-TYPE      -|                                                                 |
-TYPE       |                                                                 |
-PROMPT &CHOICE. WHAT IS YOUR CHOICE?.-----------------------------------------
-IF          &CHOICE EQ 1 GOTO CH61EXEC
-   ELSE IF &CHOICE EQ 2 GOTO CH62EXEC
-   ELSE IF &CHOICE IS 'X' GOTO EXIT
-   ELSE GOTO MENU;
-CH61EXEC
EXEC CH61EXEC
-RUN
-CRTCLEAR
-GOTO MENU
-CH62EXEC
EXEC CH62EXEC
-RUN
-CRTCLEAR
-GOTO MENU
-EXIT


*********************************************************************
*                          MODULE: CH61EXEC FOCEXEC                 *
*                          WRITTEN BY: BOB REPP                     *
*                          CALLED BY: CHGEXEC FOCEXEC               *
*                          CALLS: EDICTION FOCUS, EDICTION MASTER   *
*                          PURPOSE: UPDATES VARIABLE INFORMATION    *
*                            IN THE DATA DICTIONARY                 *
*                                                                   *
*********************************************************************
-CRTCLEAR
MODIFY FILE EDICTION
PROMPT EXEC_NAME VAR_NAME
MATCH EXEC_NAME
     ON NOMATCH REJECT
     ON MATCH CONTINUE
MATCH VAR_NAME
     ON MATCH PROMPT VAR_FORMAT VAR_DESCRIPT
     ON MATCH UPDATE VAR_FORMAT VAR_DESCRIPT
     ON NOMATCH REJECT
DATA
```

```
**********************************************************************
*                       MODULE: CH62EXEC FOCEXEC                     *
*                       WRITTEN BY: BOB REPP                         *
*                       CALLED BY:   CH6EXEC FOCEXEC                 *
*                       CALLS:   EDICTION FOCUS, EDICTION MASTER     *
*                       PURPOSE: UPDATES EXEC PURPOSE IN THE         *
*                         DATA DICTIONARY                            *
*                                                                    *
**********************************************************************
-CRTCLEAR
MODIFY FILE EDICTION
PROMPT EXEC NAME PURPOSE
MATCH EXEC NAME
     ON NOMATCH REJECT
     ON MATCH UPDATE PURPOSE
DATA


**********************************************************************
*                       MODULE: DELEXEC FOCEXEC                      *
*                       WRITTEN BY: BOB REPP                         *
*                       CALLED BY: MODIEXEC FOCEXEC                  *
*                       CALLS: DEL1EXEC,DEL2EXEC,DEL3EXEC,DEL4EXEC   *
*                       PURPOSE: EXEC DELETE MENU UNDER              *
*                         DICTIONARY MAINTENANCE                     *
*                                                                    *
**********************************************************************
-FIRST
-CRTCLEAR
-TYPE THIS IS THE DELETE MENU UNDER DICTIONARY MAINTENANCE
-TYPE
-TYPE
-TYPE
-TYPE
-TYPE |  --MAIN MENU-----------------T_______________________________
-TYPE |                                                              |
-TYPE | |---MAINTENANCE MENU---------|_______________________________
-TYPE | |                                                              |
-TYPE | | |---DELETE EXEC INFORMATION--|_______________________________
-TYPE | | |                                                              |
-TYPE | | |     DELETE ALL INFORMATION ON AN EXEC ............ <1>     |
-TYPE | | |     DELETE INFO ON A VARIABLE USED IN AN EXEC .... <2>     |
-TYPE |_| |     DELETE INFO ON A MASTER FILE USED BY AN EXEC . <3>     |
-TYPE   | |     DELETE INFORMATION ON A CALLED EXEC .......... <4>     |
-TYPE   |_|     EXIT ......................................... <X>     |
-TYPE     |                                                              |
-TYPE     |                                                              |
-PROMPT &WHICH. WHAT IS YOUR CHOICE?.------------------------------------
-IF           &WHICH EQ 1 GOTO DEL1EXEC
-   ELSE IF &WHICH EQ 2 GOTO DEL2EXEC
-   ELSE IF &WHICH EQ 3 GOTO DEL3EXEC
-   ELSE IF &WHICH EQ 4 GOTO DEL4EXEC
-   ELSE IF &WHICH IS X GOTO EXIT
-   ELSE GOTO FIRST;
-DEL1EXEC
EXEC DEL1EXEC
-RUN
-CRTCLEAR
-GOTO FIRST
-DEL2EXEC
EXEC DEL2EXEC
-RUN
-CRTCLEAR
-GOTO FIRST
-DEL3EXEC
EXEC DEL3EXEC
-RUN
-CRTCLEAR
-GOTO FIRST
```

```
-DEL4EXEC
EXEC DEL4EXEC
-RUN
-CRTCLEAR
-GOTO FIRST
-EXIT

*********************************************************************
*                      MODULE: DEL1EXEC FOCEXEC                     *
*                      WRITTEN BY: BOB REPP                         *
*                      CALLED BY: DELEXEC FOCEXEC                   *
*                      CALLS: EDICTION FOCUS, EDICTION MASTER       *
*                      PURPOSE: DELETE ALL INFORMATION IN THE       *
*                        DATA DICTIONARY ON AN EXEC                 *
*                                                                   *
*********************************************************************
-CRTCLEAR
MODIFY FILE EDICTION
PROMPT EXEC_NAME
MATCH EXEC_NAME
     ON MATCH DELETE
     ON NOMATCH REJECT
DATA

*********************************************************************
*                      MODULE: DEL2EXEC FOCEXEC                     *
*                      WRITTEN BY: BOB REPP                         *
*                      CALLED BY: DELEXEC FOCEXEC                   *
*                      CALLS: EDICTION FOCUS, EDICTION MASTER       *
*                      PURPOSE: DELETE ALL INFORMATION IN THE       *
*                        DATA DICTIONARY ON A VARIABLE USED         *
*                        IN AN EXEC                                 *
*********************************************************************
-CRTCLEAR
MODIFY FILE EDICTION
PROMPT EXEC_NAME VAR_NAME
MATCH EXEC_NAME
     ON MATCH CONTINUE
     ON NOMATCH REJECT
MATCH VAR_NAME
     ON MATCH DELETE
     ON NOMATCH REJECT
DATA

*********************************************************************
*                      MODULE: DEL3EXEC FOCEXEC                     *
*                      WRITTEN BY: BOB REPP                         *
*                      CALLED BY: DELEXEC FOCEXEC                   *
*                      CALLS: EDICTION FOCUS, EDICTION MASTER       *
*                      PURPOSE: DELETE INFORMATION IN THE           *
*                        DICTIONARY ON MASTER FILES CALLED          *
*                        BY AN EXEC                                 *
*********************************************************************
-CRTCLEAR
MODIFY FILE EDICTION
PROMPT EXEC_NAME FILE_NAME
MATCH EXEC_NAME
     ON MATCH CONTINUE
     ON NOMATCH REJECT
MATCH FILE_NAME
     ON MATCH DELETE
     ON NOMATCH REJECT
DATA
```

```
*********************************************************************
*                        MODULE: DEL4EXEC FOCEXEC                   *
*                        WRITTEN BY: BOB REPP                       *
*                        CALLED BY: DELEXEC FOCEXEC                 *
*                        CALLS: EDICTION FOCUS, EDICTION MASTER     *
*                        PURPOSE: DELETE INFORMATION IN THE         *
*                          DICTIONARY ON AN EXEC CALLED BY AN EXEC  *
*                                                                   *
*********************************************************************
-CRTCLEAR
MODIFY FILE EDICTION
PROMPT EXEC_NAME CALLED_EXEC
MATCH EXEC_NAME
     ON MATCH CONTINUE
     ON NOMATCH REJECT
MATCH CALLED_EXEC
     ON MATCH DELETE
     ON NOMATCH REJECT
DATA


*********************************************************************
*                        MODULE: MODIFILE FOCEXEC                   *
*                        WRITTEN BY: BOB REPP                       *
*                        CALLED BY:     MANTMENU FOCEXEC            *
*                        CALLS: ADDFILE, CHGFILE, DELFILE FOCEXEC   *
*                        PURPOSE: MASTER FILE MAINTENANCE MENU      *
*                                                                   *
*                                                                   *
*********************************************************************
-CRTCLEAR
-PRIME
-TYPE THIS IS THE FILE MAINTENANCE MENU FOR THE DATA DICTIONARY
-TYPE
-TYPE
-TYPE
-TYPE
-TYPE  |---MAIN MENU-----------------|________________________________
-TYPE  |                                                              |
-TYPE  | |---FILE MAINTENANCE MENU----|________________________________
-TYPE  | |                                                              |
-TYPE  | |                                                              |
-TYPE  | |             ADD FILE INFORMATION ...... <1>                  |
-TYPE  | |             UPDATE FILE INFORMATION ... <2>                  |
-TYPE  | |             DELETE FILE INFORMATION ... <3>                  |
-TYPE  |_|             EXIT (to main menu) ....... <X>                  |
-TYPE    |                                                              |
-TYPE    |______________________________________________________________|
-TYPE
-TYPE
-TYPE
-PROMPT &WHAT. WHAT IS YOUR ANSWER?.
-IF        &WHAT IS  1  GOTO ADDFILE
-  ELSE IF &WHAT IS  2  GOTO CHGFILE
-  ELSE IF &WHAT IS  3  GOTO DELFILE
-  ELSE IF &WHAT IS 'X' GOTO EXIT
-  ELSE GOTO PRIME;
-ADDFILE
EXEC ADDFILE
-RUN
-CRTCLEAR
-GOTO PRIME
-CHGFILE
EXEC CHGFILE
-RUN
-CRTCLEAR
-GOTO PRIME
-DELFILE
EXEC DELFILE
```

```
-RUN
-CRTCLEAR
-GOTO PRIME
-EXIT


**********************************************************************
*                      MODULE: ADDFILE FOCEXEC                       *
*                      WRITTEN BY: BOB REPP                          *
*                      CALLED BY: MODIFILE FOCEXEC                   *
*                      CALLS: ADD1FILE,ADD2FILE,ADD3FILE             *
*                      PURPOSE: ADD MENU UNDER DICTIONARY MAINTENANCE*
*                                                                    *
*                                                                    *
**********************************************************************
-CRTCLEAR
-TYPE THIS IS THE ADD MENU UNDER DICTIONARY MAINTENANCE
-FIRST
-TYPE
-TYPE
-TYPE
-TYPE
-TYPE ¦¯¯¯MAIN¯MENU¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¦_______________________________
-TYPE ¦                                                            ¦
-TYPE ¦ ¦¯¯¯FILE¯MAINTENANCE¯MENU¯¯¯¯¦_______________________________
-TYPE ¦ ¦                                                            ¦
-TYPE ¦ ¦ ¦¯¯¯ADD¯FILE¯INFORMATION¯¯¯¯¯¦_______________________________
-TYPE ¦ ¦ ¦                                                            ¦
-TYPE ¦ ¦ ¦       ADD NEW MASTER FILES ....................... <1>     ¦
-TYPE ¦ ¦ ¦       ADD SEGMENTS TO A MASTER FILE .............. <2>     ¦
-TYPE ¦_¦ ¦       ADD FIELDS TO A SEGMENT .................... <3>     ¦
-TYPE  ¯¦ ¦       EXIT ....................................... <X>     ¦
-TYPE   ¦_¦                                                            ¦
-TYPE    ¯¦                                                            ¦
-TYPE     ¦                                                            ¦
-TYPE     ¦____________________________________________________________¦
-TYPE
-PROMPT &CHOOSE. WHAT IS YOUR CHOICE?.
-IF         &CHOOSE EQ 1 GOTO ADD1FILE
-   ELSE IF &CHOOSE EQ 2 GOTO ADD2FILE
-   ELSE IF &CHOOSE EQ 3 GOTO ADD3FILE
-   ELSE IF &CHOOSE IS 'X' GOTO EXIT
-   ELSE GOTO FIRST;
-ADD1FILE
EXEC ADD1FILE
-RUN
-CRTCLEAR
-GOTO FIRST
-ADD2FILE
EXEC ADD2FILE
-RUN
-CRTCLEAR
-GOTO FIRST
-ADD3FILE
EXEC ADD3FILE
-RUN
-CRTCLEAR
-GOTO FIRST
-EXIT


**********************************************************************
*                      MODULE: ADD1FILE FOCEXEC                      *
*                      WRITTEN BY: BOB REPP                          *
*                      CALLED BY: ADDFILE FOCEXEC                    *
*                      CALLS: FDICTION FOCUS , FDICTION MASTER       *
*                      PURPOSE: ADD NEW MASTER FILES TO DATABASE     *
*                        DICTIONARY                                  *
*                                                                    *
**********************************************************************
```

```
-CRTCLEAR
MODIFY FILE FDICTION
PROMPT FILE_NAME
MATCH FILE_NAME
     ON MATCH REJECT
     ON NOMATCH PROMPT FILE_TYPE NUM_SEGS
     ON NOMATCH PROMPT FILE_DESCRIP MAINTAIN_BY
     ON NOMATCH INCLUDE
DATA


**********************************************************************
*                       MODULE: ADD2FILE FOCEXEC                     *
*                       WRITTEN BY: BOB REPP                         *
*                       CALLED BY: ADDFILE FOCEXEC                   *
*                       CALLS: FDICTION FOCUS, FDICTION MASTER       *
*                       PURPOSE: ADD SEGMENTS TO THE DICTIONARY      *
*                         FOR MASTER FILES IN THE DICTIONARY         *
*                                                                    *
**********************************************************************
-CRTCLEAR
MODIFY FILE FDICTION
PROMPT FILE_NAME SEG_NAME
MATCH FILE_NAME
     ON MATCH CONTINUE
     ON NOMATCH REJECT
MATCH SEG_NAME
     ON MATCH REJECT
     ON NOMATCH PROMPT CHILD_OF SEG_TYPE SEG_DESCRIPT
     ON NOMATCH INCLUDE
DATA


**********************************************************************
*                       MODULE: ADD3FILE FOCEXEC                     *
*                       WRITTEN BY: BOB REPP                         *
*                       CALLED BY: ADDFILE FOCEXEC                   *
*                       CALLS: FDICTION FOCUS, FDICTION MASTER       *
*                       PURPOSE: ADDS FIELDS TO THE DATA             *
*                         DICTIONARY FOR SEGMENTS IN THE             *
*                         DICTIONARY                                 *
**********************************************************************
-CRTCLEAR
MODIFY FILE FDICTION
PROMPT FILE_NAME SEG_NAME FIELD_NAME
MATCH FILE_NAME
     ON MATCH CONTINUE
     ON NOMATCH REJECT
MATCH SEG_NAME
     ON MATCH CONTINUE
     ON NOMATCH REJECT
MATCH FIELD_NAME
     ON MATCH REJECT
     ON NOMATCH PROMPT ALTERNATE FIELD_FORMAT FIELD_TYPE
     ON NOMATCH PROMPT KEY FIELD_DESCRI
     ON NOMATCH INCLUDE
DATA


**********************************************************************
*                       MODULE: DELFILE FOCEXEC                      *
*                       WRITTEN BY: BOB REPP                         *
*                       CALLED BY: MODIFILE FOCEXEC                  *
*                       CALLS: DEL1FILE,DEL2FILE,DEL3FILE FOCEXEC    *
*               7[C     P7RCOSE:     DELETE FILE MENU UNDER          *
*                         DICTIONARY MAINTENANCE                     *
*                                                                    *
**********************************************************************
-FIRST
```

```
-CRTCLEAR
-TYPE THIS IS THE DELETE MENU UNDER DICTIONARY MAINTENANCE
-TYPE
-TYPE
-TYPE
-TYPE
-TYPE |---MAIN-MENU----------------|_______________________________
-TYPE |                                                            |
-TYPE | |---FILE-MAINTENANCE-MENU----|_______________________________
-TYPE | |                                                            |
-TYPE | | |---ADD-FILE-INFORMATION-----|_______________________________
-TYPE | | |                                                            |
-TYPE | | |       DELETE ALL INFO ON A MASTER FILE ........... <1>     |
-TYPE | | |       DELETE ALL INFO ON A SEGMENT ............... <2>     |
-TYPE |_| |       DELETE ALL INFO ON A FIELD ................. <3>     |
-TYPE  ¯| |       EXIT ....................................... <X>     |
-TYPE   |_|                                                            |
-TYPE    ¯|                                                            |
-TYPE     |____________________________________________________________|
-TYPE
-PROMPT &WHICH. WHAT IS YOUR CHOICE?.
-IF         &WHICH EQ 1 GOTO DEL1FILE
-   ELSE IF &WHICH EQ 2 GOTO DEL2FILE
-   ELSE IF &WHICH EQ 3 GOTO DEL3FILE
-   ELSE IF &WHICH IS X GOTO EXIT
-   ELSE GOTO FIRST;
-DEL1FILE
EXEC DEL1FILE
-RUN
-CRTCLEAR
-GOTO FIRST
-DEL2FILE
EXEC DEL2FILE
-RUN
-CRTCLEAR
-GOTO FIRST
-DEL3FILE
EXEC DEL3FILE
-RUN
-CRTCLEAR
-GOTO FIRST
-EXIT


**********************************************************************
*                      MODULE: DEL1FILE FOCEXEC                      *
*                      WRITTEN BY: BOB REPP                          *
*                      CALLED BY: DELFILE FOCEXEC                    *
*                      CALLS: FDICTION FOCUS, FDICTION MASTER        *
*                      PURPOSE: DELETE ALL INFORMATION IN THE        *
*                        DATA DICTIONARY ON A MASTER FILE            *
*                                                                    *
**********************************************************************
-CRTCLEAR
MODIFY FILE FDICTION
PROMPT FILE_NAME
MATCH FILE_NAME
     ON MATCH DELETE
     ON NOMATCH REJECT
DATA
```

```
**********************************************************************
*                       MODULE: DEL2FILE FOCEXEC                     *
*                       WRITTEN BY: BOB REPP                         *
*                       CALLED BY: DELFILE FOCEXEC                   *
*                       CALLS: FDICTION FOCUS, FDICTION MASTER       *
*                       PURPOSE: DELETE INFORMATION IN THE DATA      *
*                         DICTIONARY ON A SEGMENT OF A MASTER        *
*                         FILE                                       *
**********************************************************************
-CRTCLEAR
MODIFY FILE FDICTION
PROMPT FILE_NAME SEG_NAME
MATCH FILE_NAME
     ON MATCH CONTINUE
     ON NOMATCH REJECT
MATCH SEG_NAME
     ON NOMATCH REJECT
     ON MATCH DELETE
DATA


**********************************************************************
*                       MODULE: DEL3FILE FOCEXEC                     *
*                       WRITTEN BY: BOB REPP                         *
*                       CALLED BY: DELFILE FOCEXEC                   *
*                       CALLS: FDICTION FOCUS, FDICTION MASTER       *
*                       PURPOSE: DELETE INFORMATION IN THE DATA      *
*                         DICTIONARY ON ONE FIELD IN A SEGMENT       *
*                         OF A MASTER FILE                           *
**********************************************************************
-CRTCLEAR
MODIFY FILE FDICTION
PROMPT FILE_NAME SEG_NAME FIELD_NAME
MATCH FILE_NAME
     ON NOMATCH REJECT
     ON MATCH CONTINUE
MATCH SEG_NAME
     ON NOMATCH REJECT
     ON MATCH CONTINUE
MATCH FIELD_NAME
     ON NOMATCH REJECT
     ON MATCH DELETE
DATA


**********************************************************************
*                       MODULE:   CHGFILE FOCEXEC                    *
*                       WRITTEN BY: BOB REPP                         *
*                       CALLED BY: MODIFILE FOCEXEC                  *
*                       CALLS: CHG1FILE,CHG2FILE,CHG3FILE            *
*                       PURPOSE: UPDATE FILE MENU UNDER              *
*                         DICTIONARY MAINTENANCE                     *
*                                                                    *
**********************************************************************
-SECOND
-CRTCLEAR
-TYPE THIS IS THE UPDATE MENU UNDER DICTIONARY MAINTENANCE
-TYPE
```

```
-TYPE
-TYPE
-TYPE
-TYPE ¦¯¯¯MAIN¯MENU¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¦________________________________
-TYPE ¦                                                               ¦
-TYPE ¦ ¦¯¯¯FILE¯MAINTENANCE¯MENU¯¯¯¯¦________________________________
-TYPE ¦ ¦                                                               ¦
-TYPE ¦ ¦ ¦¯¯¯UPDATE¯FILE¯INFORMATION¯¯¦________________________________
-TYPE ¦ ¦ ¦                                                               ¦
-TYPE ¦ ¦ ¦      UPDATE MASTER FILE INFORMATION ............. <1>      ¦
-TYPE ¦ ¦ ¦      UPDATE SEGMENT INFORMATION ................. <2>      ¦
-TYPE ¦_¦ ¦      UPDATE FIELD INFORMATION.................... <3>      ¦
-TYPE  ¯¦ ¦      EXIT ....................................... <X>      ¦
-TYPE   ¦_¦                                                           ¦
-TYPE    ¯¦                                                           ¦
-TYPE     ¦___________________________________________________________¦
-TYPE
-PROMPT &CHOICE. WHAT IS YOUR CHOICE?.
-IF          &CHOICE EQ 1 GOTO CHG1FILE
-   ELSE IF &CHOICE EQ 2 GOTO CHG2FILE
-   ELSE IF &CHOICE EQ 3 GOTO CHG3FILE
-   ELSE IF &CHOICE IS 'X' GOTO EXIT
-   ELSE GOTO SECOND;
-CHG1FILE
EXEC CHG1FILE
-RUN
-CRTCLEAR
-GOTO SECOND
-CHG2FILE
EXEC CHG2FILE
-RUN
-CRTCLEAR
-GOTO SECOND
-CHG3FILE
EXEC CHG3FILE
-RUN
-CRTCLEAR
-GOTO SECOND
-EXIT


*********************************************************************
*                       MODULE:   CHG1FILE FOCEXEC                  *
*                       WRITTEN BY: BOB REPP                        *
*                       CALLED BY: CHGFILE FOCEXEC                  *
*                       CALLS: FDICTION FOCUS, FDICTION MASTER      *
*                       PURPOSE: UPDATES MASTER FILE INFORMATION    *
*                         IN THE DATA DICTIONARY                    *
*                                                                   *
*********************************************************************
-CRTCLEAR
MODIFY FILE FDICTION
PROMPT FILE_NAME
MATCH FILE_NAME
     ON NOMATCH REJECT
     ON MATCH PROMPT FILE_NAME NUM_SEGS FILE_DESCRIP MAINTAIN_BY
     ON MATCH UPDATE FILE_TYPE NUM_SEGS FILE_DESCRIP MAINTAIN_BY
DATA


*********************************************************************
*                       MODULE: CHG2FILE FOCEXEC                    *
*                       WRITTEN BY: BOB REPP                        *
*                       CALLED BY: CHGFILE FOCEXEC                  *
*                       CALLS: FDICTION FOCUS, FDICTION MASTER      *
*                       PURPOSE: UPDATES SEGMENT INFORMATION        *
*                         IN THE DATA DICTIONARY                    *
*                                                                   *
*********************************************************************
```

```
-CRTCLEAR
MODIFY FILE FDICTION
PROMPT FILE_NAME SEG_NAME
MATCH FILE_NAME
     ON MATCH CONTINUE
     ON NOMATCH REJECT
MATCH SEG_NAME
     ON NOMATCH REJECT
     ON MATCH PROMPT CHILD_OF SEG_TYPE SEG_DESCRIPT
     ON MATCH UPDATE CHILD_OF SEG_TYPE SEG_DESCRIPT
DATA


***********************************************************************
*                      MODULE: CHG3FILE FOCEXEC                       *
*                      WRITTEN BY: BOB REPP                           *
*                      CALLED BY: CHGFILE FOCEXEC                     *
*                      CALLS: FDICTION FOCUS, FDICTION MASTER         *
*                      PURPOSE: UPDATES FIELD INFORMATION             *
*                        IN THE DATA DICTIONARY                       *
*                                                                     *
***********************************************************************
-CRTCLEAR
MODIFY FILE FDICTION
PROMPT FILE_NAME SEG_NAME FIELD_NAME
MATCH FILE_NAME
     ON NOMATCH REJECT
     ON MATCH CONTINUE
MATCH SEG_NAME
     ON NOMATCH REJECT
     ON MATCH CONTINUE
MATCH FIELD_NAME
     ON NOMATCH REJECT
     ON MATCH PROMPT ALTERNATE FIELD_FORMAT FIELD_TYPE KEY FIELD_DESCRI
     ON MATCH UPDATE ALTERNATE FIELD_FORMAT FIELD_TYPE KEY FIELD_DESCRI
DATA
```

```
**********************************************************************
*                      MODULE: EXECMENU FOCEXEC                      *
*                      WRITTEN BY: BOB REPP                          *
*                      CALLED BY: MAINMENU FOCEXEC                   *
*                      CALLS: EXECDESC,EXECINFO,SUMEXEC FOCEXEC      *
*                      PURPOSE: MENU FOR "CANNED" DICTIONARY         *
*                        REPORTS                                     *
*                                                                    *
**********************************************************************
-START
-CRTCLEAR
-TYPE THIS MENU ALLOWS THE USER TO ACCESS INFORMATION ON FOCUS EXECS
-TYPE
-TYPE
-TYPE
-TYPE
-TYPE
-TYPE  |---MAIN MENU-----------------|_______________________________
-TYPE  |                                                            |
-TYPE  | |---EXEC MENU-----------------|_______________________________|
-TYPE  | |                                                            |
-TYPE  | |                                                            |
-TYPE  | |             EXEC DESCRIPTIONS ......... <1>                |
-TYPE  | |             VARIABLE INFORMATION ...... <2>                |
-TYPE  | |             SUMMARY REPORT ............ <3>                |
-TYPE  |_|             EXIT (return to main menu). <X>                |
-TYPE    |                                                            |
-TYPE    |____________________________________________________________|
-TYPE
-TYPE
-PROMPT &CHOICE. WHAT IS YOUR CHOICE?.
-IF          &CHOICE EQ 1 GOTO DESCRIPT
-   ELSE IF &CHOICE EQ 2 GOTO INFO
-   ELSE IF &CHOICE EQ 3 GOTO SUMMARY
-   ELSE IF &CHOICE IS 'X' GOTO EXIT
-   ELSE GOTO START;
-DESCRIPT
EXEC EXECDESC
-RUN
-GOTO START
-INFO
EXEC EXECINFO
-RUN
-GOTO START
-SUMMARY
EXEC SUMEXEC
-RUN
-GOTO START
-EXIT


**********************************************************************
*                      MODULE: EXECINFO FOCEXEC                      *
*                      WRITTEN BY: BOB REPP                          *
*                      CALLED BY: EXECMENU FOCEXEC                   *
*                      CALLS: EDICTION FOCUS, EDICTION MASTER        *
*                      PURPOSE: PRINTS VARIABLE INFORMATION BY EXEC  *
*                        FOR EVERY EXEC IN THE DATA DICTIONARY       *
*                                                                    *
**********************************************************************
-CRTCLEAR
TABLE FILE EDICTION
PRINT VAR_NAME VAR_FORMAT VAR_DESCRIPT
BY EXEC_NAME
FOOTING CENTER
"TYPE QUIT TO RETURN TO MENU"
RUN
```

```
**********************************************************************
*                      MODULE: EXECDESC FOCEXEC                      *
*                      WRITTEN BY: BOB REPP                          *
*                      CALLED BY: EXECMENU FOCEXEC                   *
*                      CALLS: EDICTION FOCUS, EDICTION MASTER        *
*                      PURPOSE: PRINTS EXEC NAME AND PURPOSE         *
*                        FOR EVERY EXEC IN THE DATA DICTIONARY       *
*                                                                    *
**********************************************************************
-CRTCLEAR
TABLE FILE EDICTION
PRINT EXEC_NAME PURPOSE
FOOTING CENTER
"TYPE QUIT TO RETURN TO MENU"
RUN


**********************************************************************
*                      MODULE: SUMEXEC FOCEXEC                       *
*                      WRITTEN BY: BOB REPP                          *
*                      CALLED BY: EXECMENU FOCEXEC                   *
*                      CALLS: EDICTION FOCUS, EDICTION MASTER        *
*                             FDICTION FOCUS, FDICTION MASTER        *
*                      PURPOSE: JOINS FDICTION AND EDICTION IN       *
*          ORDER TO PRINT CALLED EXECS AND MASTER FILES USED         *
*          BY EVERY EXEC IN THE DATA DICTIONARY                      *
**********************************************************************
-CRTCLEAR
JOIN FILE_NAME IN EDICTION TO FILE_NAME IN FDICTION AS JJOIN
TABLE FILE EDICTION
PRINT CALLED_EXEC FILE_NAME FILE_DESCRIP
BY EXEC_NAME
FOOTING CENTER
"TYPE QUIT TO RETURN TO MENU"
RUN


**********************************************************************
*                      MODULE: FILEMENU FOCEXEC                      *
*                      WRITTEN BY: BOB REPP                          *
*                      CALLED BY: MAINMENU FOCEXEC                   *
*                      CALLS: FILDESC,SEGINFO,FLDINFO FOCEXEC        *
*                      PURPOSE: MENU OF "CANNED" REPORTS FROM        *
*                        THE DICTIONARY ON MASTER FILES              *
*                                                                    *
**********************************************************************
-START
-CRTCLEAR
-TYPE THIS MENU ALLOWS THE USER TO ACCESS INFORMATION ON FOCUS FILES
-TYPE
-TYPE
-TYPE
-TYPE
-TYPE
-TYPE  |---MAIN MENU----------------|______________________________
-TYPE  |                                                             |
-TYPE  | |---FILE MENU----------------|________________________________
-TYPE  | |                                                               |
-TYPE  | |                                                               |
-TYPE  | |            FILE DESCRIPTIONS ......... <1>                    |
-TYPE  | |            SEGMENT INFORMATION ....... <2>                    |
-TYPE  | |            FIELDNAME INFORMATION ..... <3>                    |
-TYPE  |_|            SUMMARY REPORT ............ <4>                    |
-TYPE    |            EXIT (return to main menu). <X>                    |
-TYPE    |_______________________________________________________________|
-TYPE
-TYPE
-PROMPT &CHOICE. WHAT IS YOUR CHOICE?.
-IF         &CHOICE EQ 1 GOTO DESCRIPT
```

```
-   ELSE IF &CHOICE EQ 2 GOTO SINFO
-   ELSE IF &CHOICE EQ 3 GOTO FINFO
-   ELSE IF &CHOICE EQ 4 GOTO SUMMARY
-   ELSE IF &CHOICE IS 'X' GOTO EXIT
-   ELSE GOTO START;
-DESCRIPT
EXEC FILDESC
-RUN
-CRTCLEAR
-GOTO START
-SINFO
EXEC SEGINFO
-RUN
-CRTCLEAR
-GOTO START
-FINFO
EXEC FLDINFO
-RUN
-CRTCLEAR
-GOTO START
-SUMMARY
EXEC SUMFILE
-RUN
-CRTCLEAR
-GOTO START
-EXIT

**********************************************************************
*                       MODULE: FILDESC FOCEXEC                      *
*                       WRITTEN BY: BOB REPP                         *
*                       CALLED BY: FILEMENU FOCEXEC                  *
*                       CALLS: FDICTION FOCUS, FDICTION MASTER       *
*                       PURPOSE: PRINTS FILE NAME, DESCRIPTION, ETC. *
*                         ON EVERY MASTER FILE IN THE DICTIONARY     *
*                                                                    *
**********************************************************************
-CRTCLEAR
TABLE FILE FDICTION
PRINT FILE_NAME FILE_DESCRIP MAINTAIN_BY FILE_TYPE NUM_SEGS
FOOTING CENTER
"TYPE QUIT TO RETURN TO MENU"
RUN

**********************************************************************
*                       MODULE: FLDINFO FOCEXEC                      *
*                       WRITTEN BY: BOB REPP                         *
*                       CALLED BY: FILEMENU FOCEXEC                  *
*                       CALLS: FDICTION FOCUS, FDICTION MASTER       *
*                       PURPOSE: PRINTS FIELD NAME, DESCRIPTION,     *
*                         AND ALIAS BY MASTER FILE FOR EVERY         *
*                         FIELD IN THE DATA DICTIONARY               *
**********************************************************************
-CRTCLEAR
TABLE FILE FDICTION
PRINT FIELD_NAME FIELD_DESCRI ALTERNATE
BY FILE_NAME
FOOTING CENTER
"TYPE QUIT TO RETURN TO MENU"
RUN
```

```
*********************************************************************
*                      MODULE: SEGINFO FOCEXEC                      *
*                      WRITTEN BY: BOB REPP                         *
*                      CALLED BY: FILEMENU FOCEXEC                  *
*                      CALLS: FDICTION FOCUS, FDICTION MASTER       *
*                      PURPOSE:    PRINT SEGMENT NAME, DESCRIPTION  *
*                        AND PARENT BY MASTER FILE FOR EVERY        *
*                        SEGMENT IN THE DICTIONARY                  *
*********************************************************************
-CRTCLEAR
TABLE FILE FDICTION
PRINT SEG_NAME SEG_DESCRIPT CHILD_OF SEG_TYPE
BY FILE_NAME
FOOTING CENTER
"TYPE QUIT TO RETURN TO MENU"
RUN


*********************************************************************
*                      MODULE: SUMFILE FOCEXEC                      *
*                      WRITTEN BY: BOB REPP                         *
*                      CALLED BY: FILEMENU FOCEXEC                  *
*                      CALLS: FDICTION FOCUS, FDICTION MASTER       *
*                      PURPOSE: PRINTS FILE, SEGMENT, AND FIELD     *
*                        INFORMATION FOR EVERY MASTER FILE IN       *
*                        THE DATA DICTIONARY                        *
*********************************************************************
-CRTCLEAR
TABLE FILE FDICTION
PRINT FIELD_NAME ALTERNATE KEY  FIELD_TYPE
BY FILE_NAME
BY SEG_NAME
FOOTING CENTER
"TYPE QUIT TO RETURN TO MENU"
RUN
```

SAMPLE DICTIONARY REPORTS

```
FILE_NAME  FILE_DESCRIP                 MAINTAIN_BY    FILE_TYPE  NUM_SEGS
---------  ------------                 -----------    ---------  --------
FDICTION   MASTER FILE DICTIONARY       DESIGNER       FOC               3
EDICTION   EXEC MASTER DICTIONARY       R.C.REPP       FOC               4
ADMISSIO   STUD AND COURSES TAKEN       NAV ACAD       FOC               2
AVAIL      CREDIT HRS BY COURSE         NAV ACAD       FOC               1
REQ        COURSE SUBJ AREAS            NPS            FOC               1
```

```
FILE_NAME  SEG_NAME  SEG_DESCRIPT                CHILD_OF  SEG_TYPE
---------  --------  ------------                --------  --------
ADMISSIO   COURSE    STUD GRADES BY COURSE       STUDENT   S1
           STUDENT   STUD ID AND INFO                      S1
AVAIL      DESCRIPT  CREDIT HRS BY COURSE        ACOURSE   S1
EDICTION   CALLED    EXECS CALLED BY AN EXEC     EXECS     S1
           EXECS     FOCEXEC DESCRIPTIONS        EDICTION  S1
           FILE_NAM  MASTER FILES USED           EXECS     S1
           VARIABLE  VARIABLE INFORMATION        EXECS     S1
FDICTION   FIELDS    INFO ON SEGMENT FIELDS      SEGMENTS  S1
           FILES     INFO ON MASTER FILES        FDICTION  S1
           SEGMENTS  INFO ON MASTER SEGMENTS     FILES     S1
REQ        DESCRIPT  COURSE SUBJ AREAS           RCOURSE   S1
```

```
FILE_NAME  FIELD_NAME    FIELD_DESCRI                ALTERNATE
---------  ----------    ------------                ---------
ADMISSIO   COURSE_ID     COURSE ID NUMBER            CID
           LETTER_GRADE  FINAL GRADE ASSIGNED        LG
           SEMESTER      WHEN TAKEN (SEM/YEAR)       WHEN
           MAJOR         AREA OF DEGREE              MAJ
           NATIONALITY   COUNTRY OF ORIGIN           RACE
           SEX           SEX                         SEX
           SOC_SEC_NUM   SOCIAL SECURITY NO.         SSN
           STUD_ID       STUDENT ID                  SID
           STUD_NAME     STUDENT NAME                SN
AVAIL      COURSE_ID     COURSE ID NUMBER            CID
           CREDIT_HOURS  0.0 TO 4.0 COURSE CREDIT    CHRS
EDICTION   CALLED_EXEC   NAME OF CALLED EXEC         CE
           EXEC_NAME     NAME OF FOCEXEC             EN
           PURPOSE       PURPOSE OF FOCEXEC          DOES
           FILE_NAME     FOCUS MASTER FILE DESCRIP   FLN
           VAR_DESCRIPT  DESCRIPTION OF VARIABLE     VDES
           VAR_FORMAT    ALLOWABLE VARIABLE FORMAT   VFMT
           VAR_NAME      VARIABLES USED IN FOCEXEC   VN
FDICTION   ALTERNATE     ALIAS                       ALT
           FIELD_DESCRI  INFO ON SEGMENT FIELDS      FDDE
           FIELD_FORMAT  TYPE & AMOUNT OF DATA       FFMT
           FIELD_NAME    NAME OF FIELD               FDN
           FIELD_TYPE    I MEANS INDEXED             FDTP
           KEY           Y MEANS KEY FIELD           KEY
           FILE_DESCRIP  DESCRIPTION OF MASTER       FDES
           FILE_NAME     NAME OF FOCUS MASTER FILE   FLN
           FILE_TYPE     TYPE OF MASTER FILE         FTYP
           MAINTAIN_BY   ACTIVITY RESPONSIBLE        FMAN
           NUM_SEGS      NUMBER OF SEGMENTS          NS
           CHILD_OF      PARENT OF SEGMENT           SPAR
           SEG_DESCRIPT  INFO ON MASTER SEGMENTS     SDES
           SEG_NAME      NAME OF MASTER SEGMENT      SEGN
           SEG_TYPE      SEGMENT KEY & ORDER INFO    STYP
REQ        COURSE_ID     COURSE ID NUMBER            CID
           SUBJ_AREA     SUBJECT AREA OF COURSE      SA
```

```
FILE_NAME  SEG_NAME  FIELD_NAME    ALTERNATE  KEY  FIELD_TYPE
---------  --------  ----------    ---------  ---  ----------
ADMISSIO   COURSE    COURSE_ID     CID        Y
                     LETTER_GRADE  LG         N
                     SEMESTER      WHEN       N
           STUDENT   MAJOR         MAJ        N
                     NATIONALITY   RACE       N
                     SEX           SEX        N
                     SOC_SEC_NUM   SSN        N
                     STUD_ID       SID        Y
                     STUD_NAME     SN         N
AVAIL      DESCRIPT  COURSE_ID     CID        Y    I
                     CREDIT_HOURS  CHRS       N
EDICTION   CALLED    CALLED_EXEC   CE         Y
           EXECS     EXEC_NAME     EN         Y
                     PURPOSE       DOES       N
           FILE_NAM  FILE_NAME     FLN        Y    I
           VARIABLE  VAR_DESCRIPT  VDES       N
                     VAR_FORMAT    VFMT       N
                     VAR_NAME      VN         Y
FDICTION   FIELDS    ALTERNATE     ALT        N
                     FIELD_DESCRI  FDDE       N
                     FIELD_FORMAT  FFMT       N
                     FIELD_NAME    FDN        Y
                     FIELD_TYPE    FDTP       N
                     KEY           KEY        N
           FILES     FILE_DESCRIP  FDES       N
                     FILE_NAME     FLN        Y    I
                     FILE_TYPE     FTYP       N
                     MAINTAIN_BY   FMAN       N
                     NUM_SEGS      NS         N
           SEGMENTS  CHILD_OF      SPAR       N
                     SEG_DESCRIPT  SDES       N
                     SEG_NAME      SEGN       Y
                     SEG_TYPE      STYP       N
REQ        DESCRIPT  COURSE_ID     CID        Y    I
                     SUBJ_AREA     SA         N


EXEC_NAME  PURPOSE
---------  -------
MAINMENU   DATA DICTIONARY CHOICES
FILEMENU   FOCUS FILE INFO MENU
EXECMENU   FOCUS EXECS INFO MENU
MANTMENU   MAINTENANCE MENU
MODIEXEC   MODIFY EXEC INFO MENU
CHGEXEC    UPDATE EXEC INFO MENU
CHG1EXEC   UPDATES VARIABLE INFO
DELEXEC    DELETE EXEC INFO MENU
DEL1EXEC   DELETE INFO ON AN EXEC
DEL2EXEC   DELETE INFO ON A VARIABLE
DEL3EXEC   DELETE INFO ON A FILE
DEL4EXEC   DELETE INFO ON EXEC
ADDEXEC    ADD EXEC INFORMATION
ADD1EXEC   ADD EXEC NAMES
ADD2EXEC   ADD VARIABLE INFO
ADD3EXEC   ADD MASTER FILE INFO
ADD4EXEC   ADD CALLED EXEC INFO
CHG2EXEC   UPDATES EXEC PURPOSE
PROJMENU   CALLS PROGRAM OR DICTION
CALCQPR    CALCULATES QPR BY STUDENT
SUBJSUM    TABLES GRADES BY SUBJ
```

```
EXEC_NAME  VAR_NAME  VAR_FORMAT  VAR_DESCRIPT
---------  --------  ----------  ------------
ADDEXEC    &CHOOSE               MENU CHOICE
CALCQPR    NGRADE    F3.1        LETTER GRADE TO NUM GRADE
           POINTS    F6.1        NGRADE * CHRS
           QPR       F4.1        POINTS/CHRS
CHGEXEC    &CHOICE               MENU SELECTION
           &YESNO                Y/N TO CONTINUE
EXECMENU   &CHOICE               MENU CHOICE
           &OK                   OK TO CONTINUE Y/N
FILEMENU   &CHOICE               MENU CHOICE
           &OK                   OK TO CONTINUE Y/N
MAINMENU   &SELECT               MENU CHOICE
MANTMENU   &OK                   OK TO CONTINUE Y/N
           &WHICH                MENU CHOICE
MODIEXEC   &WHAT                 MENU CHOICE
PROJMENU   &SELECT               MENU CHOICE
```

```
EXEC_NAME  CALLED_EXEC  FILE_NAME  FILE_DESCRIP
---------  -----------  ---------  ------------
ADDEXEC    ADD1EXEC     .          .
           ADD2EXEC     .          .
           ADD3EXEC     .          .
           ADD4EXEC     .          .
ADD1EXEC   .            EDICTION   EXEC MASTER DICTIONARY
ADD2EXEC   .            EDICTION   EXEC MASTER DICTIONARY
ADD3EXEC   .            EDICTION   EXEC MASTER DICTIONARY
ADD4EXEC   .            EDICTION   EXEC MASTER DICTIONARY
CALCQPR    PROJMENU     ADMISSIO   STUD AND COURSES TAKEN
           .            AVAIL      CREDIT HRS BY COURSE
CHGEXEC    CHG1EXEC     .          .
CHG1EXEC   .            EDICTION   EXEC MASTER DICTIONARY
DELEXEC    DEL1EXEC     .          .
           DEL2EXEC     .          .
           DEL3EXEC     .          .
           DEL4EXEC     .          .
DEL1EXEC   .            EDICTION   EXEC MASTER DICTIONARY
DEL2EXEC   .            EDICTION   EXEC MASTER DICTIONARY
DEL3EXEC   .            EDICTION   EXEC MASTER DICTIONARY
DEL4EXEC   .            EDICTION   EXEC MASTER DICTIONARY
EXECMENU   EXECDESC     .          .
           EXECINFO     .          .
           SUMEXEC      .          .
FILEMENU   FILDESC      .          .
           FLDINFO      .          .
           SUMFILE      .          .
MAINMENU   EXECMENU     .          .
           FILEMENU     .          .
           MANTMENU     .          .
MANTMENU   MODIEXEC     .          .
           MODIFILE     .          .
MODIEXEC   ADDEXEC      .          .
           CHGEXEC      .          .
           DELEXEC      .          .
PROJMENU                .          .
SUBJSUM    PROJMENU     ADMISSIO   STUD AND COURSES TAKEN
           .            REQ        COURSE SUBJ AREAS
```

APPENDIX B

TRANSCRIPT SUMMARY APPLICATION SOURCE CODE

```
**********************************************************************
*                      MODULE: PROJMENU FOCEXEC                      *
*                      WRITTEN BY: BOB REPP                          *
*                      CALLED BY:                                    *
*                      CALLS: CALCQPR, SUBJSUM, MAINMENU FOCEXEC     *
*                      PURPOSE: MAIN PROJECT MENU                    *
*                                                                    *
*                                                                    *
**********************************************************************
SET MSG=OFF
-CRTCLEAR
-BEGIN
-CRTCLEAR
-TYPE
-TYPE
-TYPE
-TYPE
-TYPE
-TYPE
-TYPE   |---PROJECT MENU-------------|
-TYPE   |                             ---------------------------------|
-TYPE   |                                                              |
-TYPE   |     STUDENT TRANSCRIPT SUMMARY ...... <1>                    |
-TYPE   |     DATA DICTIONARY ................. <2>                    |
-TYPE   |     QUIT ............................ <Q>                    |
-TYPE   |                                                              |
-TYPE   |                                                              |
-TYPE   |--------------------------------------------------------------|
-TYPE
-TYPE
-TYPE
-PROMPT &SELECT. WHAT IS YOUR CHOICE?.
-IF          &SELECT EQ 1 GOTO CALCQPR
-   ELSE IF &SELECT EQ 2 GOTO MAINMENU
-   ELSE IF &SELECT IS 'Q' GOTO EXIT
-   ELSE GOTO BEGIN;
-CALCQPR
EXEC CALCQPR
-RUN
EXEC SUBJSUM
-RUN
-CRTCLEAR
-GOTO BEGIN
-MAINMENU
EXEC MAINMENU
-RUN
-CRTCLEAR
-GOTO BEGIN
-EXIT
```

```
***********************************************************************
*                       MODULE: CALCQPR FOCEXEC                       *
*                       WRITTEN BY: BOB REPP                          *
*                       CALLED BY: PROJMENU FOCEXEC                   *
*                       CALLS: ADMISSIO FOCUS, AVAIL FOCUS            *
*                              AVAIL MASTER, ADMISSIO MASTER          *
*                       PURPOSE: JOINS FILES AND CALCULATES QPR       *
*       FOR EACH STUDENT BASED ON ALL COURSES TAKEN EXCEPT THOSE      *
*       WITH A GRADE OF "V"                                           *
***********************************************************************
-CRTCLEAR
JOIN CID IN ADMISSIO TO CID IN AVAIL AS AJOIN
DEFINE FILE ADMISSIO
NGRADE/F3.1=IF LG EQ 'A' THEN 4.0
  ELSE IF LG EQ 'B' THEN 3.0
  ELSE IF LG EQ 'C' THEN 2.0
  ELSE IF LG EQ 'D' THEN 1.0
  ELSE 0.0;

POINTS/F6.1=NGRADE*CHRS;
END
TABLE FILE ADMISSIO
SUM POINTS NOPRINT CHRS NOPRINT
COMPUTE QPR/F4.1=POINTS/CHRS;
BY SID BY STUD_NAME BY SSN BY MAJOR
IF LG OMITS 'V'
END


***********************************************************************
*                       MODULE: SUBJSUM FOCEXEC                       *
*                       WRITTEN BY: BOB REPP                          *
*                       CALLED BY: PROJMENU FOCEXEC                   *
*                       CALLS:ADMISSIO FOCUS,ADMISSIO MASTER          *
*                             REQ FOCUS, REQ MASTER                   *
*                       PURPOSE: JOINS ADMISSIO AND REQ AND PRINTS    *
*       THE NUMBER OF COURSES TAKEN BY SUBJECT AREA ACROSS GRADES     *
***********************************************************************
JOIN CID IN ADMISSIO TO CID IN REQ AS BJOIN
TABLE FILE ADMISSIO
FOOTING CENTER
"TYPE QUIT TO RETURN TO MENU"
COUNT CID ACROSS LG
BY SID BY SA
ON SID PAGE-BREAK
RUN
```

TRANSCRIPT SUMMARY APPLICATION OUTPUT

```
 STUD_ID  STUD_NAME          SOC_SEC_NUM  MAJOR   QPR
 -------  ---------          -----------  -----   ---
 840007   ABBOT LOHN F       423788586    ESE     3.7
 840019   ABELL DAVID GROS   512727258    SAS     2.4
 840024   ABRAMSON ALAN J    037367483    SPH     3.0


                    LETTER_GRADE
                       A       B       C       D       V
 STUD_ID  SUBJ_AREA
 -----------------------------------------------------------
 840007   MATH_C       0       1       0       0       0
          MATH_PO      1       0       0       0       0
          MATH_PR      1       0       0       0       1
          STATS        0       0       1       0       0
```

```
                        LETTER_GRADE
                           A         B          C          D          V
STUD_ID   SUBJ_AREA
-------------------------------------------------------------------------
840019    CHEM              1         0          0          0          0
          MATH_C            0         0          1          0          0
          MATH_PO           0         0          1          0          0
          PHY_CD            0         0          1          0          0
          PHY_UD            0         0          0          1          0


                        LETTER_GRADE
                           A         B          C          D          V
STUD_ID   SUBJ_AREA
-------------------------------------------------------------------------
840024    A_M_ENG           0         1          0          0          0
          ELECT_E           0         1          0          0          0
          MATH_PR           0         0          0          0          1
          OPHYSCI           0         1          0          0          0
          OTH_ENG           0         1          0          0          0
```

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Superintendent Attn: Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Director of Admissions, Code 0145 Naval Postgraduate School Monterey, California 93943-5100	2
4.	Professor D. R. Dolk, Code 54Dk Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5100	1
5.	Professor N. R. Lyons, Code 54Lb Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5100	1
6.	LCDR R. C. Repp, USN Patrol Squadron FOUR FPO San Francisco, CA. 96601-5901	1
7.	Computer Tedhnology Programs Code 37 Naval Postgraduate School Monterey, California 93943-5100	1

602 - 197

215143

Thesis
R358 Repp
c.1 Evaluation of a data dictionary during application program design based upon the FOCUS DEMS.